COMENTARIO TECNICO



Herramientas de Hard & Soft.

2da y última Parte.

Herramientas de Software.

Hasta aquí, se han visto algunas de las herramientas disponibles de hardware para trabajar con la familia HC908, pero nos está faltando el software para completar nuestro "arsenal" de combate. Hoy en día se dispone de "Entornos Integrados de Trabajo" conocidos como **IDE** (**I**ntegrated **D**evelopment **E**nvironment) que contienen en un solo lugar o "entorno" todos los aplicativos necesarios para trabajar en forma "profesional" con los microcontroladores de la familia HC908. Si bien existen una infinidad de "IDEs" para los HC908, los más populares son el ***"WinIDE"*** de la firma P & E Microcomputer Systems y el ***"CodeWarrior"*** de Freescale Semiconductor. Ambos sistemas corren bajo entorno ***"Windows"*** pero poseen distintas prestaciones y características que los hacen útiles en distintos momentos de nuestro aprendizaje.

Por ejemplo, el sistema ***"WinIDE"*** es muy sencillo de usar, muy intuitivo, con simples botones "gráficos" se realizan tareas de borrado / grabación de la memoria FLASH, compilación, emulación, simulación, etc., pero tanta "simplicidad" paga el precio de la poca flexibilidad a la hora de crecer con las prestaciones y la compatibilidad con otras familias de 8 y 32 bits de Freescale. Mientras que en el sistema ***"CodeWarrior"*** encontramos un ambiente de trabajo muy profesional, apto para trabajar con lenguajes assembler, "C", "C++" y con gran flexibilidad para soportar distintas familias de MCUs, herramientas de hardware, etc., pero paga el precio de ser un ambiente algo más "complicado" y menos "intuitivo" que el ***WinIDE***, por lo que no es recomendable su uso para personas que recién se inician en el mundo de los microcontroladores y si para aquellos que ya tienen algún camino recorrido en el mismo.

A continuación veremos como es trabajar con cada uno de ellos.

Trabajando con el entorno "WinIDE".

Como la mejor forma de aprender a usar una herramienta es "usándola", veremos aquí un ejemplo práctico de uso con el sistema didáctico ***"EDUKIT08".***

Una vez instalado el entorno ***WinIDE*** correspondiente al MCU contenido en la placa **"PLUGIN_AP"** incluida en el kit (el MCU es el MC908AP32CFBE, ver la instalación de dicho entorno en el **"Manual de Usuario"** del sistema ***EDUKIT08*** contenido en el CD ROM de instalación del mismo), se dispondrá de una serie de iconos (archivos ejecutables) dentro de la solapa principal del mismo **(Ver figura 1).**

Se deberá correr primero el archivo WINIDE.EXE (también se podrá acceder al mismo por medio de la implementación de un "acceso directo" al escritorio de trabajo, **ver figura 2.**), el mismo es un programa editor y funciona como un lanzador (shell) para otros módulos (WinIDE Development Environment) **(ver Figuras 3 y 4).**

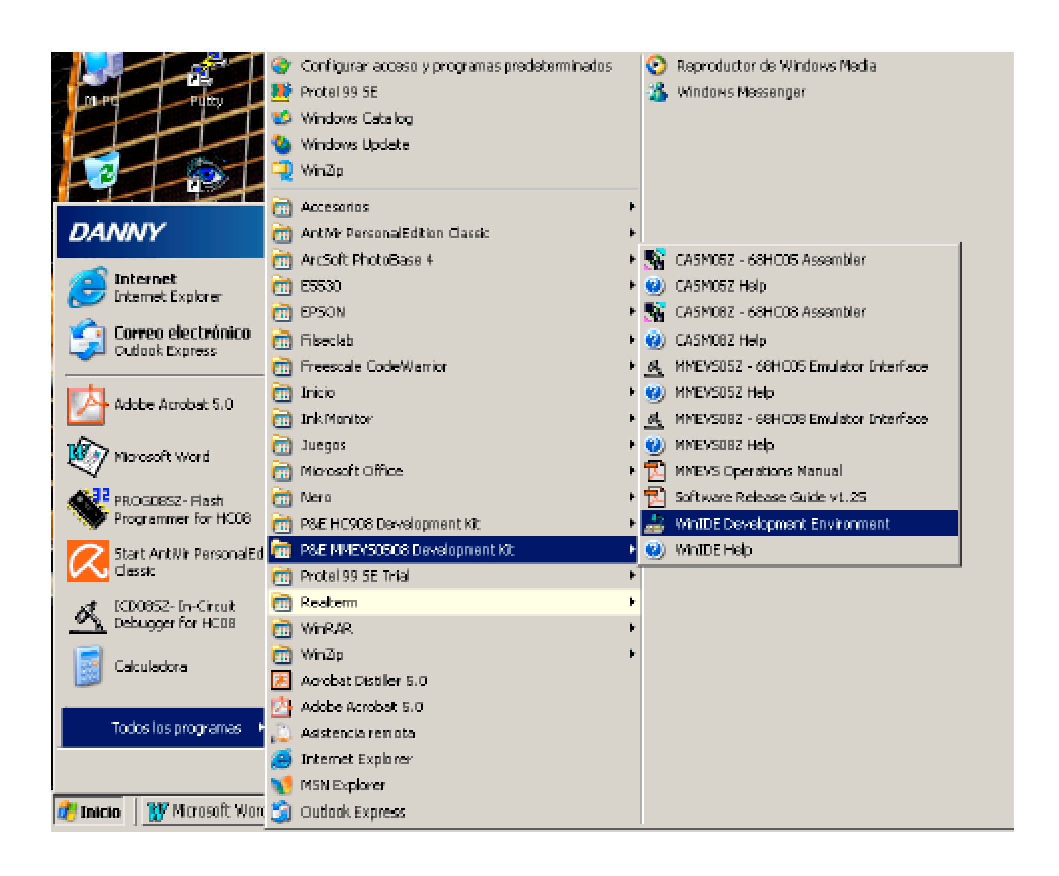


Figura 1.- Solapas del entorno WinIDE.



Figura 2.- Icono de acceso directo en el escritorio de trabajo al entorno WinIDE.

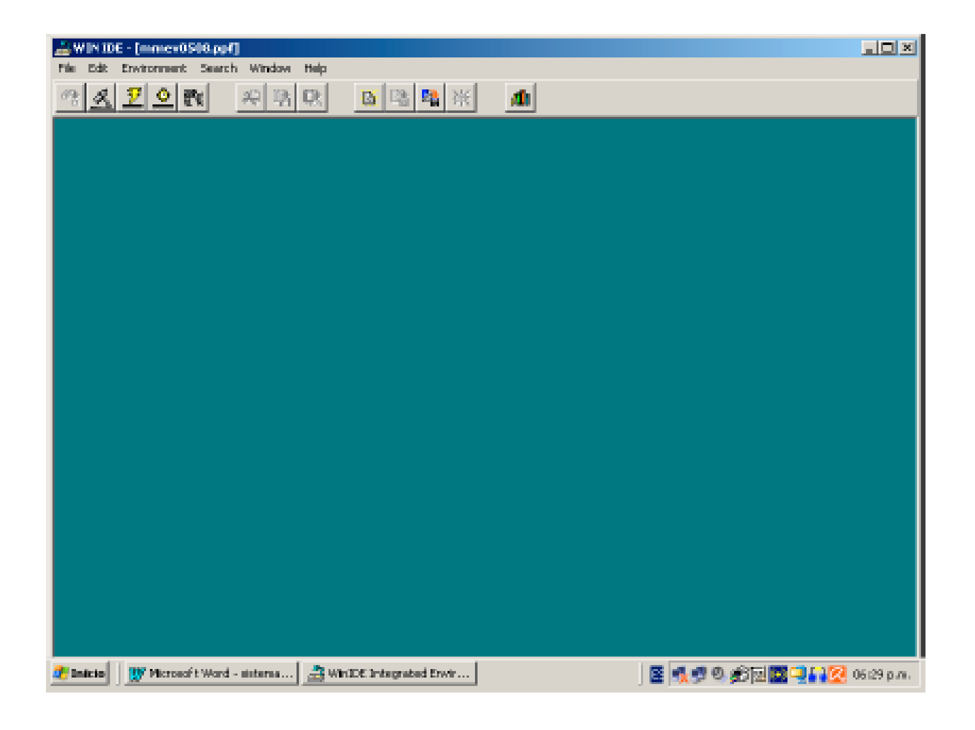
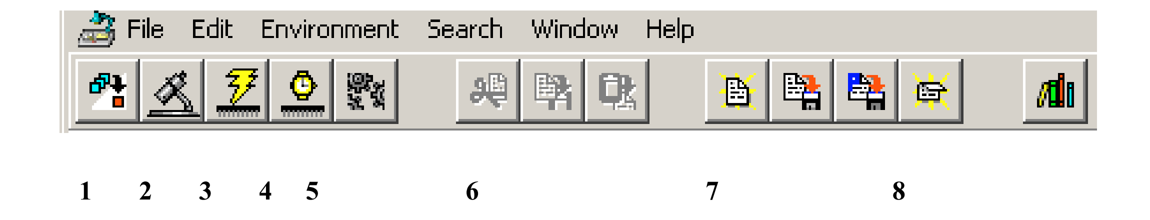


Figura 3.- Pantalla principal y de edición del entorno WinIDE.



1 2 3 4 5 6 7 8

1 – Compilador / Ensamblador (CASM08SZ)
2 – Simulador En – Circuito (ICS08SZ)
3 – Programador (PROG08SZ)
4 – Emulador en Tiempo Real (ICD08SZ)
5 - Simulador Puro (ICS08SZ)
6 – Botones de Edición (Copy / Paste / Cut)
7 – Botones de manejo de archivos (Open / Close / Save File / Save Project)
8 - Archivos de Registros

Figura 4.- Barra de Herramientas e iconos de aplicaciones del entorno.

En el entorno **WinIDE** instalado para el sistema ***EDUKIT08*** no se dispone de los siguientes aplicativos:

- **Simulador Puro (sin circuito) (ICS08SZ).**
- **Simulador En – Circuito (ICS08SZ).**
- **Archivos de Registros para los simuladores.**

Esto se debe a que la "simulación" se basa en correr el programa del usuario en la PC y no en el MCU propiamente dicho, como se ha visto en detalle en los primeros capítulos de este curso. Ello obliga a disponer de un "programa" o aplicativo de simulación para cada uno de los MCUs de la familia HC908. Lamentablemente, se disponen de muchos simuladores para los primeros dispositivos de la familia HC908, pero no así para la familia 908APxx que es la que posee la placa PLUGIN_AP incluida en el kit.

La "Simulación En - Circuito" y la "Simulación Pura" son de relativa utilidad ya que no involucran hechos "reales", en "tiempos reales", y pueden hacer pensar al usuario que lo "simulado" sea la "realidad", cuando nada más lejos de ello.

Hechas las aclaraciones del caso, comenzaremos a trabajar en nuestro ejemplo práctico, el archivo aquí elegido lleva el nombre **"TEMP01.ASM"** y tiene por objeto mostrar el funcionamiento del módulo A/D en resolución de 10 bits, conectado a un sensor de temperatura (LM335Z) y mostrar lo medido por medio del display 7 segmentos de 4 dígitos LEDs que posee el sistema didáctico.

Para ello, procederemos a configurar los "jumpers" del sistema según nuestras necesidades y de acuerdo a los explicado en **"Manual del Usuario"** del sistema ***EDUKIT08*** (como por ejemplo, ***"guía rápida de uso"***, o ***"Asignación de Jumpers"***).

Que para nuestro ejemplo será:

JP1 ---- Placa "PLUGIN_AP" ---- **Posición 2-3** (oscilador externo 20 Mhz).

JP2A / JP2B / JP2C --- Placa Principal

Posición 1-2 ---- Uso del Puerto Serial RS-232C "CN2" al COMx de la PC.
(usar fuente de alimentación Externa!!).

Posición 2-3 ---- Uso del Puerto Serial USB "CN1" al puerto USB 2.0 de la PC.
(No usar fuente de alimentación Externa!!).

JP3 --- Placa Principal ---- **Posición 1-2** (Control de alimentación por DTR).

JP4 --- Placa Principal ---- **Posición 2-3** (+VHIGH en pin RESET)

JP5 --- Placa Principal ---- **Posición Cerrado** (Manejo del pin de Reset)

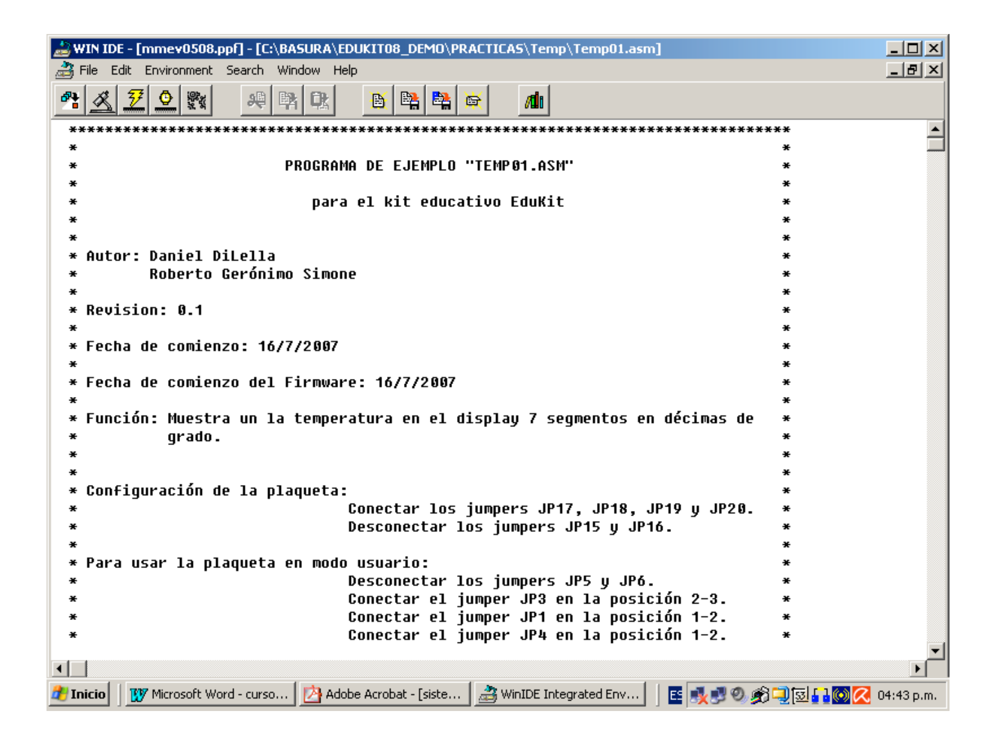
JP6 --- Placa Principal ---- **Posición Cerrado** (Manejo del pin de IRQ)

JP15 / JP16 --- Placa Principal ---- **Posición Abierto** (Display LCD OFF).

JP17/JP18/JP19/JP20 --- Placa Principal ---- **Posición Cerrado** (Display 7 Segmentos LEDs Activo).

JP21 --- Placa Principal ---- **Posición 1-2** (Sensor Temperatura activo en A/D).

Una vez comprobada la configuración y hecha la conexión entre el sistema didáctico y la PC por medio del cable serial RS-232C (y fuente de alimentación) o por medio del cable serial USB (no usar fuente de alimentación!!) se procederá a abrir el programa de práctica como se muestra en la siguiente figura (por medio de la opción **File / Open**, o el botón **"Open File"** de la barra de Herramientas).....



El archivo **“TEMP01.ASM”** es solo un archivo de tipo texto que puede ser modificado por el usuario para realizar numerosas pruebas por medio del editor de texto y sus herramientas de edición, presentes desde el primer momento de la ejecución del entorno WinIDE.
El próximo paso que necesita el **archivo ASM**, es ser ensamblado, creando un **código objeto en formato S19** (formato Motorola) para ”bajarlo” al HC908AP32 de la placa “PLUGIN_AP” incorporada en el sistema para luego realizar trabajos de emulación en Tiempo Real en circuito.
Esta tarea se realiza haciendo un click en el botón “**Compiler / Assembler**” **(CASM08Z)** en la barra de herramientas que dispone el WinIDE. Si se encuentra un error (detectado por el compilador), el editor resaltará la línea de código conteniendo el error y detendrá la compilación, según muestra la siguiente figura.

```
*************************************************************
* Init_Timer - Turns on timer 1 Channel 1 for an Output     *
*              Compare in approximately 2ms. The timer      *
*              interrupt service routine continually sets   *
*              the next interrupt to occur 2ms later.       *
*************************************************************
Init_Timer:
       mova   #$36,TSC      ; Timer 1 - Cleared + Stopped.
                            ;    Clicks once every 64 BUS Cycles
                           ;    2000t Clicks ~ 2ms

       mov    #$0,TCH1H    ; Set Output Compare to happen 77T clicks
       mov    #$4D,TCH1L   ; after we start the timer. (~2ms). The
                            ; timer interrupt will set OC for another ~2ms.

       mov    #$54,TSC1    ; Timer 1 Channel 1
                            ; Set for Output Compare operation.

       mov    #$06,TSC      ; Start the timer
       rts
```

Pantalla típica de error durante la compilación de un programa.

El proceso se repetirá hasta que no se encuentre error alguno, y el archivo así generado quedará listo para grabarse en la memoria FLASH del dispositivo HC908AP32 para luego realizar tareas de Emulación en Tiempo Real. El compilador incluido en el entorno WinIDE **(CASM08Z)** es del tipo general o sea **sirve para cualquier HC908** sin importar tamaño de memoria de programa o configuración de periféricos existente y no contempla “linkeadores” que permitan compilar distintos “trozos” de programa en forma separada. Como resultado del proceso de compilación del archivo de texto con extensión**“.asm”** se obtienen **5 archivos en total** y ellos son los siguientes:

- Archivo con extensión **.asm** ----- **TEMP01.ASM** (archivo de texto en lenguaje assembler).

- Archivo con extensión **.bak** ------ **TEMP01.BAK** (archivo back up del archivo .asm).

- Archivo con extensión **.lst** ----- **TEMP01.LST** (archivo "list", listado de variables y etiquetas).

- Archivo con extensión **.map** ----- **TEMP01.MAP** (archivo "map" con direcciones de variables y etiquetas).

- Archivo con extensión **.s19** ----- **TEMP01.S19** (archivo "s19" con código ejecutable en formato Motorola)

Usando la función "Programación".

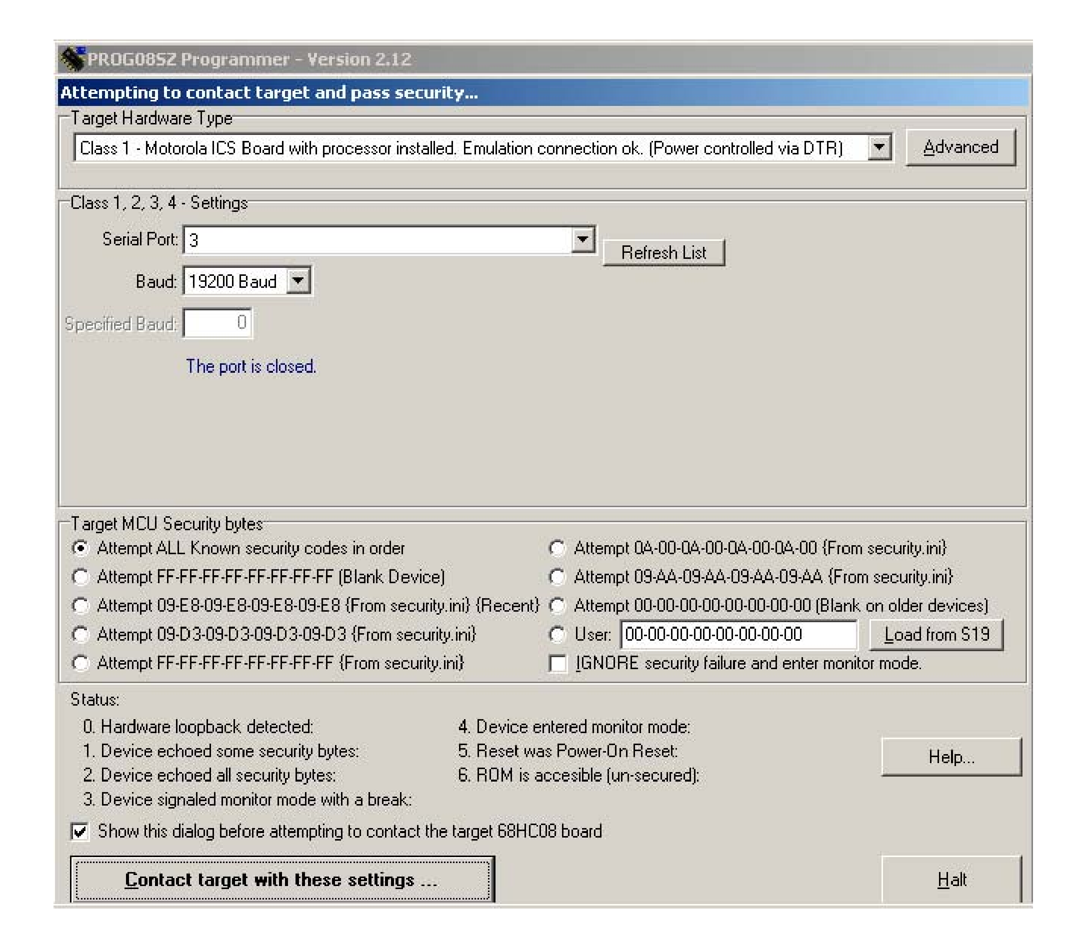
Cuando se selecciona el módulo de programación "Programmer" **(PROG08SZ)** por medio del icono de la barra de herramientas del WinIDE (o cuando el aplicativo ***PROG08SZ*** es elegido como programa principal), y **previamente se energiza el sistema por medio de una fuente externa o por medio del cable USB**, se estaría en condiciones de establecer la comunicación con el sistema ***"EDUKIT08"*** y como consecuencia de ello, se desplegará una pantalla gris de configuraciones que nos permitirá setear velocidad de comunicación y otros parámetros en función de lo establecido en las placas del sistema.

Para nuestro ejemplo utilizaremos los siguientes valores para dicha pantalla:

Target Hardware Type: CLASS I, placa con control de alimentación por señal DTR.

Serial Port: Aquí poner el Número de **puerto Serial COM Real o Virtual** (si se usa el cable USB)utilizado por el usuario en la PC.

Baud: 19200 bps. Aquí deberemos poner el Baud que tendrá directa relación con la frecuencia de Bus (FBUS) y con la frecuencia del oscilador externo elegido a inyectar por el pin OSC1 **(FXTAL = 20.000 Mhz -– FBUS = 5 Mhz).**



Target MCU Security Bytes: Attemp ALL Know Security codes in order.

En este punto se deberá tener en cuenta que en los microcontroladores de la familia HC908 existe un mecanismo de seguridad **"anti – lectura no autorizada"** que funciona en forma **"implícita"** en ellos.
Todos los HC908 están protegidos en forma automática contra lectura sin la necesidad de que el usuario tenga que habilitar dicha función, como sucedía con el antiguo esquema de protección por "bit" de seguridad implementado en los MCU OTP o de otros fabricantes, en donde el usuario tiene la responsabilidad de habilitar la función anti – lectura por medio de la escritura de un bit de seguridad (en muchos casos, del tipo irreversible) anterior o posteriormente al proceso de grabación de la memoria de programa del MCU.
El mecanismo de los HC908 está basado en un **"Security Password"** o palabra de seguridad de una longitud de 8 bytes de largo (2 elevado a la 64 posibles combinaciones !!!) que se encuentra en la memoria FLASH de los HC908 compartiendo la zona de vectores de estos, como se ha visto en detalle en capítulos anteriores.

De esta forma, la palabra de seguridad está directamente relacionada con el contenido de los vectores de interrupción utilizados por el usuario durante el diseño del programa de aplicación de este. Cada programa, al utilizar en forma distinta los vectores de interrupción, tendrá su propia palabra de seguridad válida, sin que para ello, medie alguna acción del usuario.
Se sugiere consultar el manual de datos de cualquier HC908 en la sección **"Monitor ROM"** para conocer mayores detalles del mecanismo de seguridad contra lecturas no autorizadas implementados en ellos.
Para nuestro ejemplo, se eligió la opción de intentar todos los password conocidos (memorizados en los intentos previos) aunque podríamos utilizar el password $FF FF FF FF FF FF FF FF pues es casi seguro que nuestro microcontrolador HC908AP32 de la placa PLUGIN_AP **estará con su memoria FLASH virgen** o sea **totalmente borrada ($FF)** y el password para esta condición es el anteriormente mencionado.

Status:

En este cuadro de estados se puede verificar las distintas etapas de la conexión entre el microcontrolador de la placa PLUGIN y el sistema formado por PC – EDUKIT08 (configuraciones de jumpers), entregándonos datos de en cual etapa se produce el error en la comunicación. Esto es muy útil para corregir problemas de hardware por pines mal conectados o tensiones incorrectas en las señales necesarias para asegurar el modo monitor del HC908.
Si no hubiera errores, no tendríamos retorno de este cuadro, ya que se pasaría directamente a la próxima etapa del modo programación.

Tildado del cuadro "Show this dialog before attempting to contact the target 68HC908 Board" :

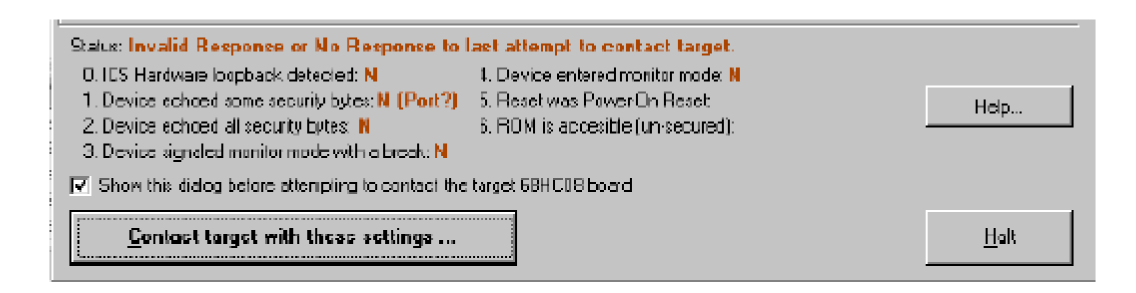
Esta opción debe ser seleccionada **(tildar el cuadro pequeño)** antes de establecer el primer contacto con el sistema ***EDUKIT08***, ya que como expresa el enunciado en inglés, **al tildar esta opción siempre se desplegará el cuadro de seteo antes de efectuar el contacto con el sistema.**

Botón de Contacto con el Sistema "Contact Target with this Settings":

Una vez completado todo el seteo de las opciones presentadas en el cuadro, el usuario debe hacer click en el cuadro de contacto para establecer efectivamente la comunicación con el sistema y se efectúa un primer intento por pasar el mecanismo de seguridad.
En este estado, se efectuará un **"ciclado de la alimentación del MCU bajo desarrollo"** en forma **totalmente automática** entre la placa y la PC **(indicado por LED 3 "System Ready" , LED 3 – ON – MCU Power – On , LED 3 – OFF – MCU Power - Off),** y dependiendo del estado del sistema esto puede ser efectuado múltiples veces.

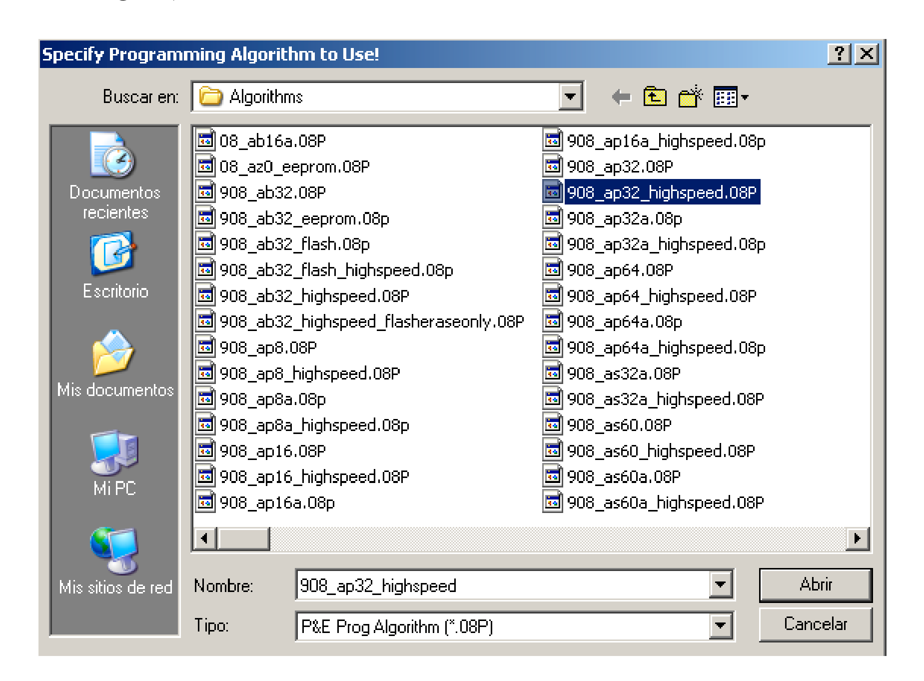
Cada vez que se remueva la alimentación de la placa (por medio de la remoción de la fuente externa o de la desconexión del cable serial USB), **se debe esperar por lo menos 1 segundo** para reanudar la alimentación, de esta forma se asegurará que el MCU efectúe su ciclo de "Power On Reset".

Si la comunicación es exitosa se pasará a la próxima pantalla del programador, **pero si no lo es**, se mostrará la pantalla explicada pero con la sección **"Status"** con los estados resultantes de cada una de las etapas de la comunicación para poder determinar en donde se ha producido el error. Se sugiere verificar configuraciones de jumpers, puertos seriales elegidos, alimentación (si se usara fuente externa y puerto RS-232C), fusible de la placa, configuración de la pantalla de comunicación, etc.).



Detalle de sección "Status" cuando hay errores en la comunicación MCU – PC.

Después de pasar el mecanismo de seguridad, se mostrará al usuario un cuadro de elección del **"algoritmo" de programación** (contiene rutinas especiales para programar al dispositivo elegido).



Seleccionar el algoritmo adecuado para el MCU bajo programación, **existe un algoritmo para cada tipo de HC908**, en nuestro sistema, deberemos elegir el **"908_ap32_Highspeed.08p",** que es el que corresponde al **MCHC908AP32CFBE** que se encuentra en la placa "PLUGIN_AP" para nuestras prácticas.

De esta forma el dispositivo ha sido inicializado y está listo para su borrado y posterior reprogramación.

Seguridad en los HC908.

Si por distintas circunstancias, no se pudiera superar el mecanismo de seguridad en forma exitosa, referirse a la **sección 10.4 del manual de Datos Técnicos referencia** del dispositivo HC908AP32, para conocer como trabaja el mismo.
El mecanismo de seguridad implementado en el MCU, previene todo intento de observar el contenido de la memoria Flash, por parte de personas extrañas al programa.

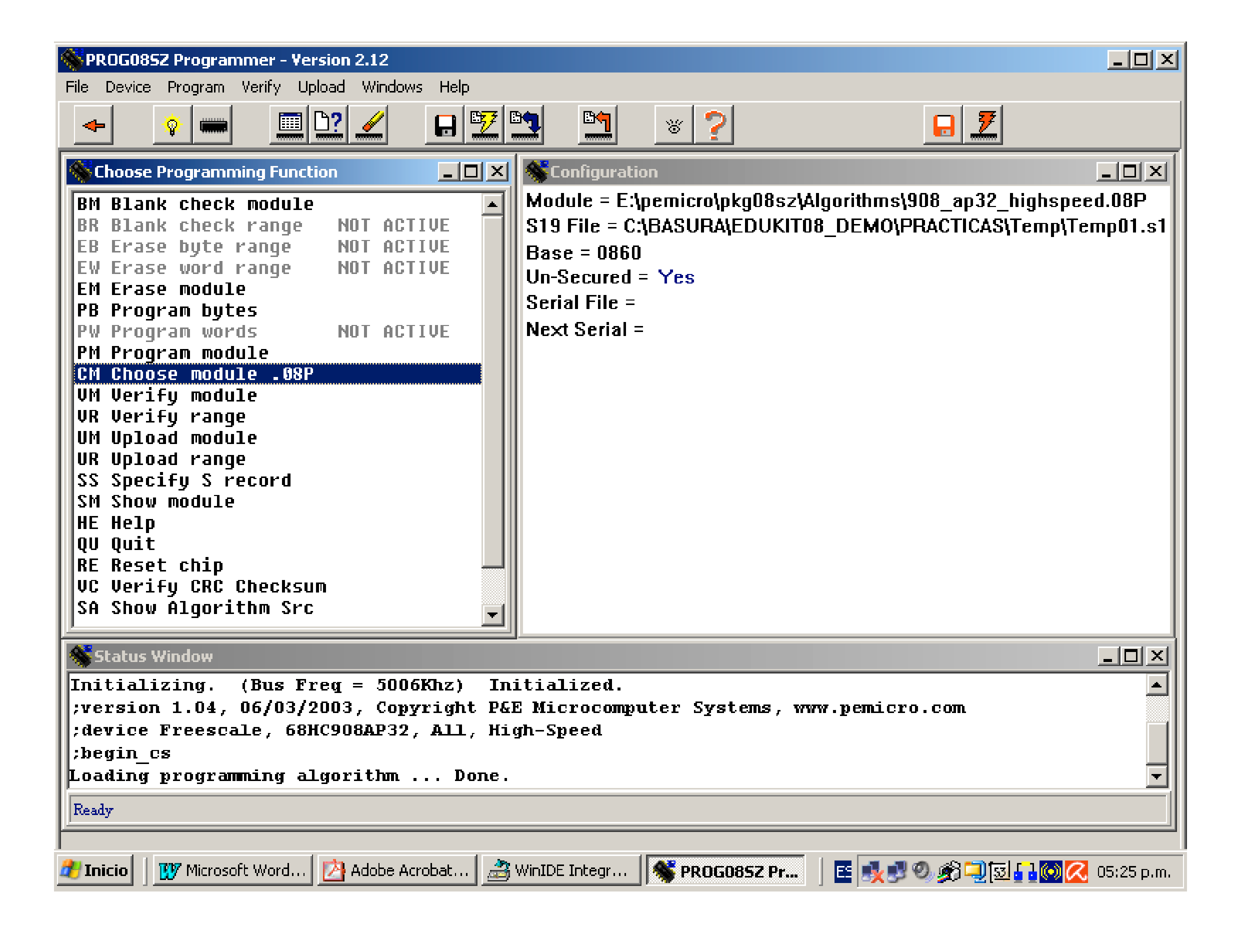
Solo el autor del programa puede acceder a todas las funciones del modo "monitor", quedando solo disponible para las personas no autorizadas la función "Borrado total de la memoria FLASH del MCU" **(Mass Erase),** que permite reutilizar el microcontrolador protegido, pero no permite la lectura de código interno en la memoria FLASH de este, por parte de personas no autorizadas.

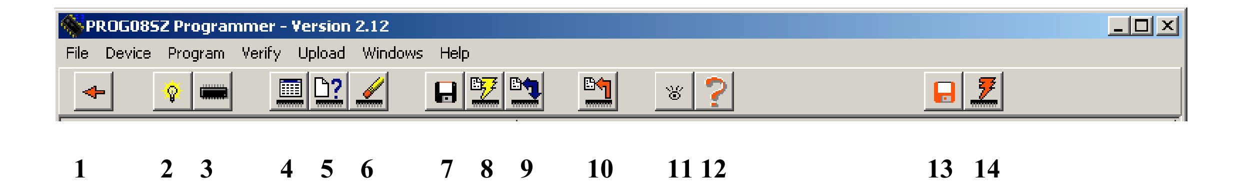
En el caso que luego de múltiples intentos fallidos por pasar la seguridad terminen sin éxito, se tiene la opción de saltar este paso y entrar al MCU al modo monitor pero en el **modo "seguro" (**o sea, no permite observar contenido alguno de memoria ni ejecutar comandos) por medio de una ventana que indicará ello **(tildar la opción IGNORE security failure and enter monitor mode).**

De esta forma solo se tendrá acceso al borrado total del dispositivo, para luego (con el mismo totalmente borrado) proceder por medio del WinIDE a la grabación de este.

Dentro del PROG08SZ “Programmer”.

Una vez superado el cuadro de comunicaciones y elegido el “algoritmo” de programación correspondiente al MCU de la placa PLUGIN (HC908AP), se ingresará a la ventana del programador ***“PROG08SZ”*** como se puede ver en la figura......





1 – Botón Volver al Editor
2 - Botón "Reset Processor"
3 – Botón "Choose Module (elegir algoritmo)
4 – Botón "View Module Data" (ver data en Flash)
5 – Botón "Black Check" (chequeo de memoria en Blanco)
6 – Botón "Erase Module" (Mass Erase – Borrado gral.)
7 – Botón "Load .S19" (carga el archivo S19 a grabar)
8 – Botón "Program Module" (Programación de la Flash)
9 - Botón "Verify Module" (verificación de lo grabado)
10 – Botón "UpLoad Module" (levanta código de la flash)
11 – Botón "Show Algoritm" (muestra el código del algoritmo)
12 – Botón "Help" (Ayuda)
13 – Botón "Specify Serial Num" (no usado aquí)
14 – Botón "Program Serial Num" (no usado aquí).

Barra de Herramientas del PROG08SZ y los distintos íconos que la componen

Lo primero que uno debería hacer es verificar si la memoria Flash del MCU está borrada (blank), para ello se procede a hacer "click" en el botón "Blank Check" de la barra de herramientas. Si ello no es así, se precederá a borra la misma haciendo "click" en el Botón "Erase Module" para efectuar un borrado gral. de la memoria Flash del MCU.

Una vez borrada la memoria, se cargará el programa "TEMP01.S19" por medio del botón "Load .S19" y se examinarán las carpetas hasta hallar la ubicación del mismo.

Se podrán ver las rutas de acceso al archivo de "algoritmo" de grabación y al archivo ".S19" a grabar en el cuadro superior derecho (Configurations) de la pantalla del programador.

Para efectuar la grabación de la memoria Flash en forma efectiva, se deberá hacer "click" en el botón "Program Module" (Rayito) y luego de unos instantes la memoria estará grabada con el archivo correspondiente. Verificar la grabación en Flash con el archivo contenido en la PC por medio del botón "Verify Module".

El programador "PROG08SZ" posee más funciones que escapan al contenido de este artículo, pero el lector puede ampliar sus conocimientos simplemente haciendo "click" en el botón de Ayuda (Help) que contiene la barra de herramientas del mismo.

Al finalizar con éxito la grabación de la memoria Flash, se deberá salir del programador haciendo "click" en la solapa "file" y luego "Exit" o simplemente haciendo "click" en la "X" superior derecha típica de las aplicaciones "Windows".

ICD08SZ In-Circuit Debugger.

La aplicación ***ICD08SZ*** **"In – Circuit Debugger"** (Depuración En – Circuito), nos permite depurar un programa en Tiempo Real o también conocida como "Emulación en Tiempo Real". La Emulación en Tiempo Real, se lleva a cabo en la familia HC908 por medio del protocolo "MON08" como se ha visto en capítulos anteriores. Este "Monitor" posee ciertas características y limitaciones que el usuario debe tener presentes cuando lo utiliza para así obtener de él, el máximo provecho y funcionalidad.

MON08, Limitaciones en el Debugging.

El debugger emplea un set de comandos que le permiten realizar **debugging en Tiempo Real** con las limitaciones del programa monitor **MON08** contenido en los dispositivos MCHC908's .

Estas limitaciones son inherentes al **"MON08"** y deben tenerse en cuenta:

1. **No debe cambiarse el bit 0** del registro Data Direction o el Data Value del PORTx (donde "x" Puede ser "A" o "B") (1-Wire Communication, ver el Data Book del dispositivo específico). Cuando se escribe, este bit debe ser seteado a "0", o sea debe configurarse el PTA0 o PTB0 (según corresponda) como entrada (INPUT) para no interferir con la comunicación entre el MCU y la PC.

2. No habilitar Keyboard Interrupts (KBI) para el bit 0 del PORTx (donde "x" puede ser "A" o "B") por igual razón que en el punto anterior.

3. No hacer "step" (paso a paso) en una instrucción que salta a si misma.

4. No hacer "step" (paso a paso) de un Software Interrupt Instruction (SWI).

5. Los registros de Hardware Breakpoint están reservados para ser usados por el **ICD08SZ debugger.**
   El uso de estos registros para otros propósitos puede NO funcionar.

6. Tener cuidado acerca de la observación de los estados de los periféricos y registros de datos en la ventana de variables o memoria. El refresco de estas ventanas leerán estos registros y podrán causar el borrado de los flags correspondientes.

7. El monitor para debug integrado dentro de los procesadores HC08, usa hasta 13 bytes del stack. No escribir desde la dirección (SP-13) a SP (Cargar un programa en la RAM).
   Ver cuidadosamente la RAM disponible para cada dispositivo específico.

8. Si las interrupciones son habilitadas durante un "stepping", el ICD08SZ debugger no hará el "step" dentro de la interrupción. En lugar de ello, se ejecutará toda la interrupción y se detendrá en la instrucción de retorno después de la interrupción.

Los siguientes puntos, son útiles de tener en cuenta:

1. Se permite "Single Stepping" tanto en RAM como en FLASH.

2. El primer Breakpoint seteado es siempre un "Hardware Breakpoint" , y cualquier breakpoint adicional será tomado como Software Breakpoint. Para esta seguro de que un Hardware Breakpoint está siendo seteado, usar el comando NOBR antes de setear el mismo. El hardware breakpoint detendrá la ejecución en FLASH y RAM.
   El software breakpoint SOLO detendrá la ejecución en RAM.

3. Ejecutar una instrucción SWI mientras se corre un programa (dentro del programa) es funcionalmente equivalente a activar un breakpoint, excepto que la ejecución se detendrá en la instrucción siguiente al SWI.

4. Para debug desde FLASH y ver el código fuente mientras se hace un "stepping", usar el comando LOADMAP. Este carga la información del código fuente contenido en los archivos de formato de archivos MAP.

5. Si un comando GO es activado sin setear previamente un breakpoint, la única forma de tomar control del procesador nuevamente, es resetearlo (usar el comando RESET).

6. El sistema Watchdog no está activo mientras se corre el ICD08SZ. Cuando el dispositivo es programado y energizado sin el debugger (EDUKIT08 en plena EMULACION), el watchdog está activado por default.
   Para inicializar y usar el ICD08SZ, por favor lea cuidadosamente la sección 7 **(ICD08SZ - In – Circuit Debugger**) y más secciones del **M68ICS08 USER'S MANUAL (WinIDE User Manual).**

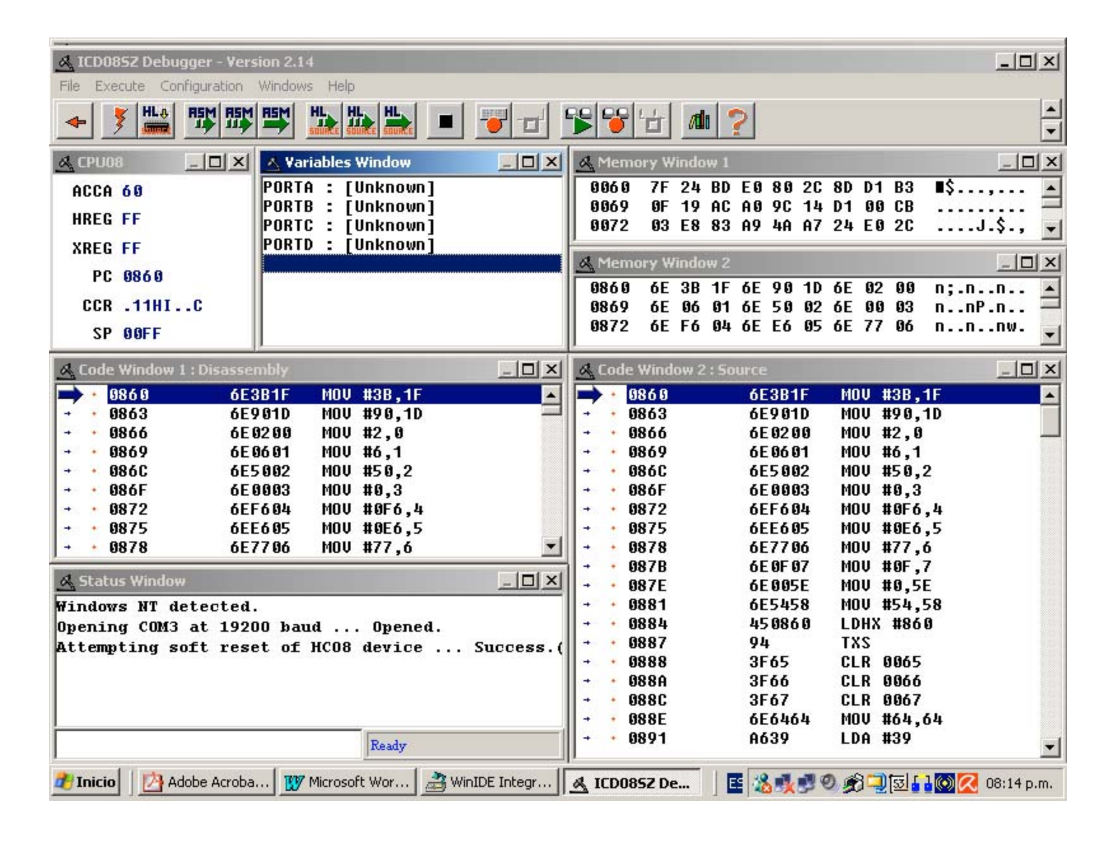
Ahora volviendo a nuestro ejemplo.......

Para hacer uso de la Emulación en Tiempo Real por medio de la aplicación **ICD08SZ** (In – Circuit Debugger) y una vez compilado nuestro programa de demostración **“TEMP01.asm”** con el compilador CASM08Z disponible en la barra de herramientas y luego de corregido todos los errores de compilación, si los hubiera, se generarán 5 archivos vitales para nuestra depuración de código.

En la sección “Usando la función Programación” se vio como programar la memoria FLASH de un dispositivo HC908**, la programación de la memoria FLASH es necesaria** cuando se quiere volcar el código del programa del usuario **en el dispositivo final** (microcontrolador de la aplicación) como paso previo al uso del **Emulador En Tiempo Real (ICD08SZ)** que nos permitirá una depuración de código en **TIEMPO REAL** con las ventajas que ello implica.

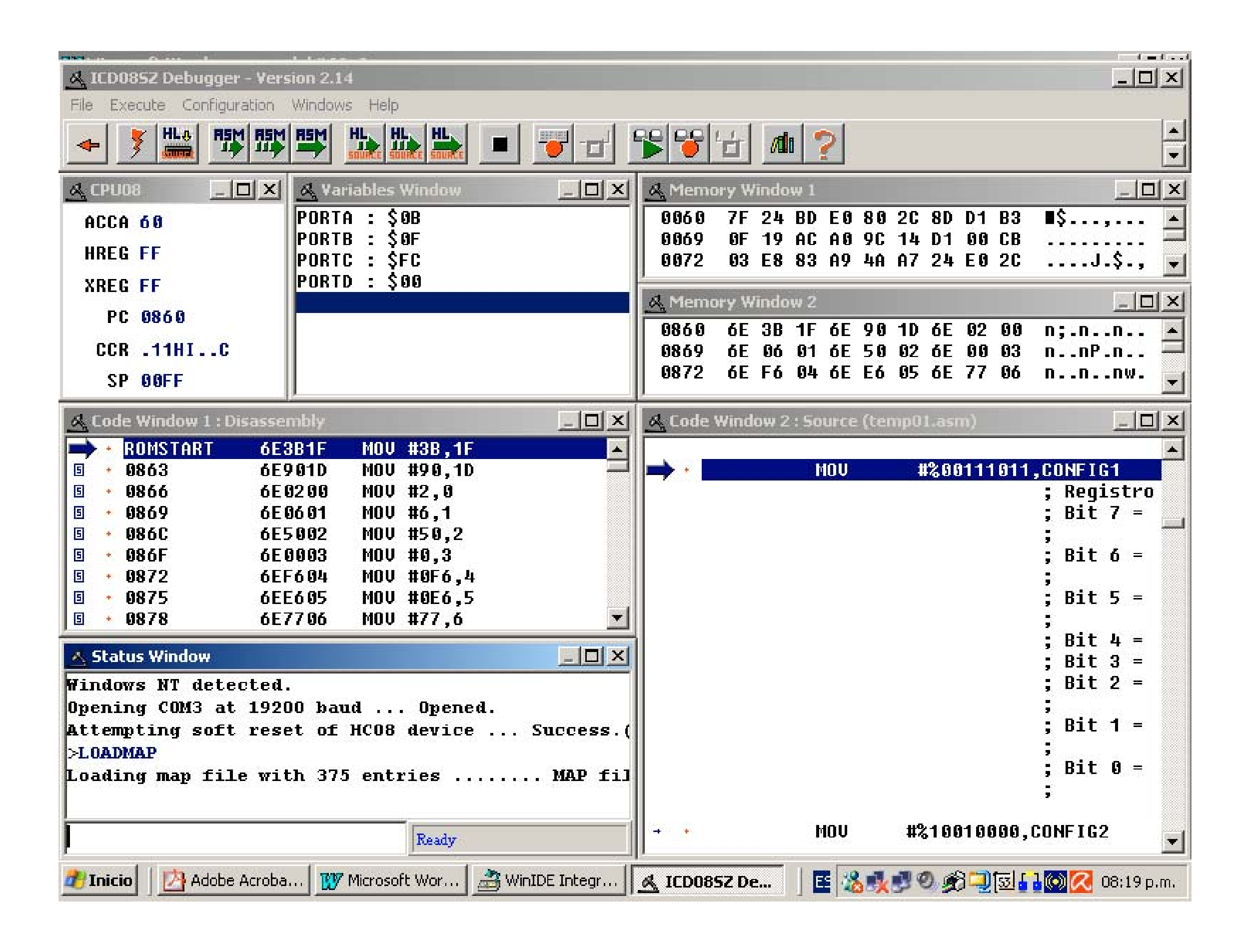
Luego de realizada la programación del MC908AP32 se podrá hacer uso del **ICD08SZ** por medio del icono presente en la barra de herramientas (**icono con forma de reloj pulsera),** como toda aplicación que necesite comunicación con la placa de hardware (**EDUKIT08**) se abrirá el cuadro de comunicación para configurar tipo de hardware, Baud Rate y otros detalles necesarios para establecer la comunicación (copiar los mismos valores de configuración a los usados durante la programación).

Superada este cuadro se nos abrirá la pantalla principal del **In Circuit Debugger** (Emulador en Tiempo Real) como se muestra en la **figura......**



Como puede observarse en la ventana ***"Code Windows 2"*** , el código fuente aparece sin comentarios o etiquetas o sea un disassembler rústico, debido a que el **ICD08SZ** no "carga" en forma automática el archivo con extensión **".MAP"** necesario para mostrar con detalles variables y etiquetas del código **".asm"** original.

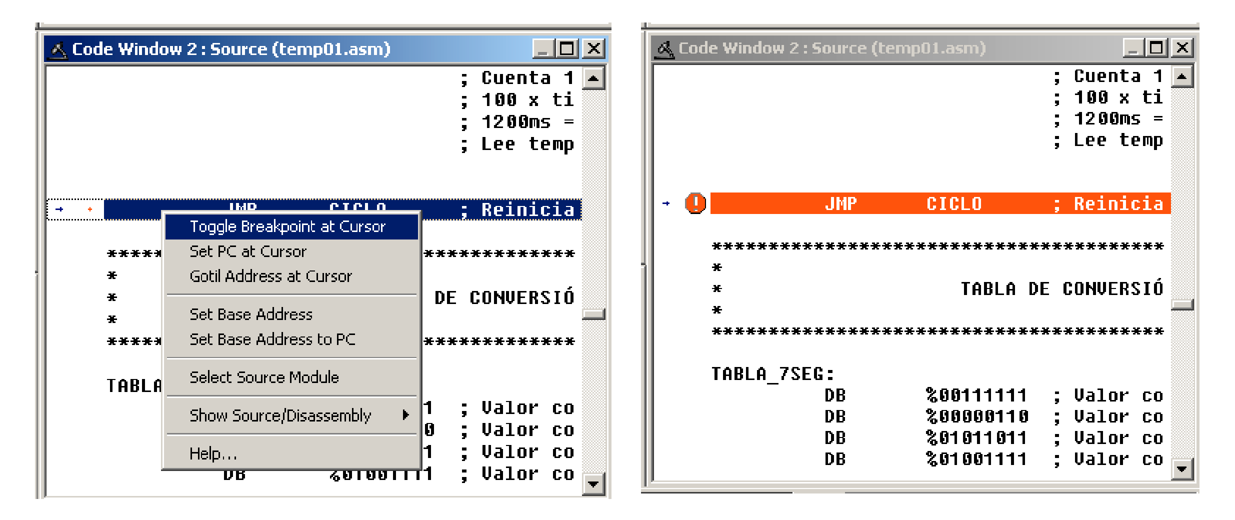
Para solucionar ello, deberemos tipear el comando **"LOADMAP"** en el cuadro de comandos o bien hacer click en el icono **"HL"** y se nos abrirá una ventana con todos los archivos **".map"** disponibles. A continuación elegimos el archivo indicado (TEMP01.map) y podremos ver la pantalla de la figura.....



Pantalla de Emulación con el archivo fuente "TEMP01.asm".

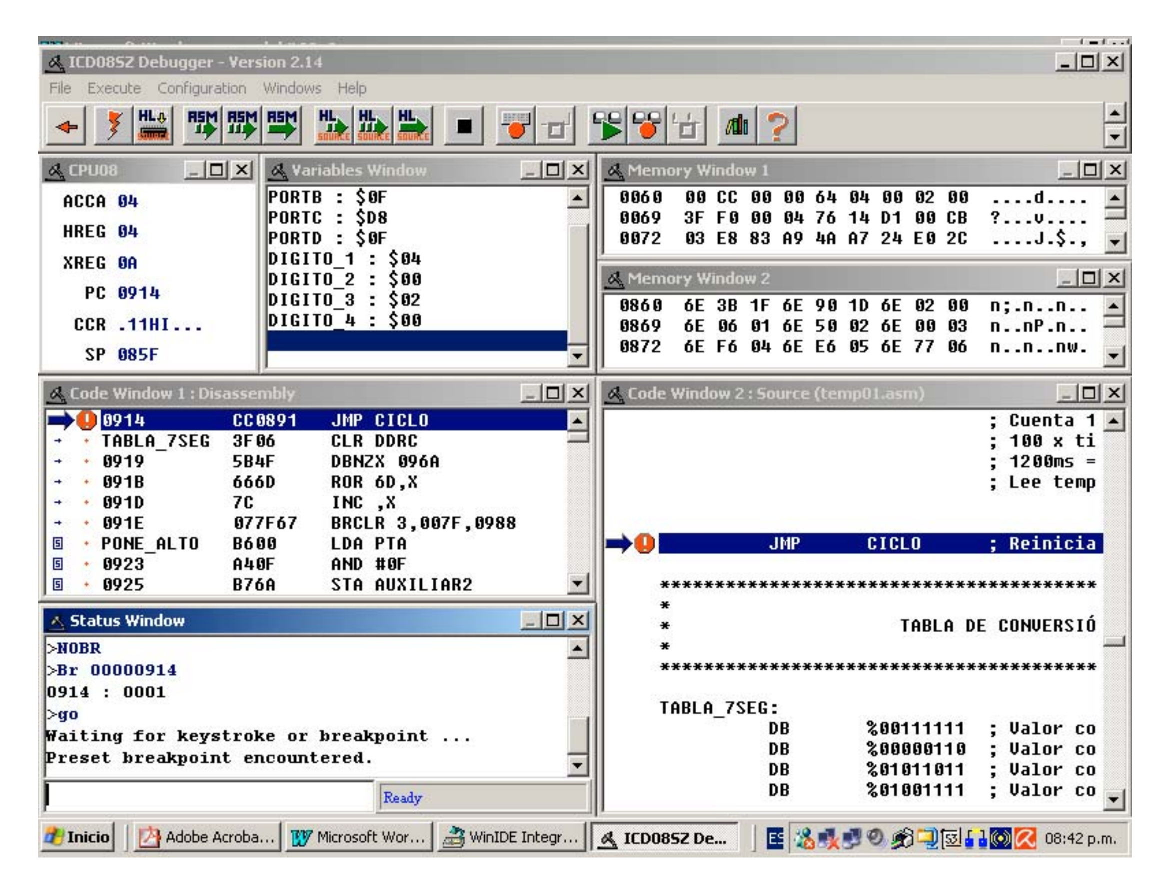
Ya estamos listos para empezar a correr nuestra "demo" en tiempo real y luego de configurar las variables a mostrar en la ventana **"Variables Window"** nos aprestaremos a colocar un **punto de parada** o **Break Point** en algún lugar del programa que nos sea de interés.
En este punto es bueno aclarar que, en la **Emulación en tiempo Real**, solo podremos usar un **SOLO break point** por hardware, ya que es el disponible en el interior del microcontrolador (ver manual de datos del HC908AP o el Reference Manual de la familia), pues solo en la simulación se dispone de más de un break point porque la misma corre en la PC y no en el MCU como aquí.

Colocamos el Break Point en una línea del programa de nuestro interés, donde queramos rescatar valores de variables y registros en ese punto exacto, de forma muy sencilla haciendo "click" en el botón "derecho" del mouse sobre la línea de interés previamente escogida, de esta forma se nos abrirá una ventana (pop up) con distintas opciones, entre ellas la opción "Toggle Breakpoint at Cursor", haciendo un simple click sobre esta opción se nos **iluminará en rojo** la línea elegida, indicándonos de esta forma que el Breakpoint ha sido aceptado y el programa está listo para comenzar a "correr" en el modo de "Emulación".

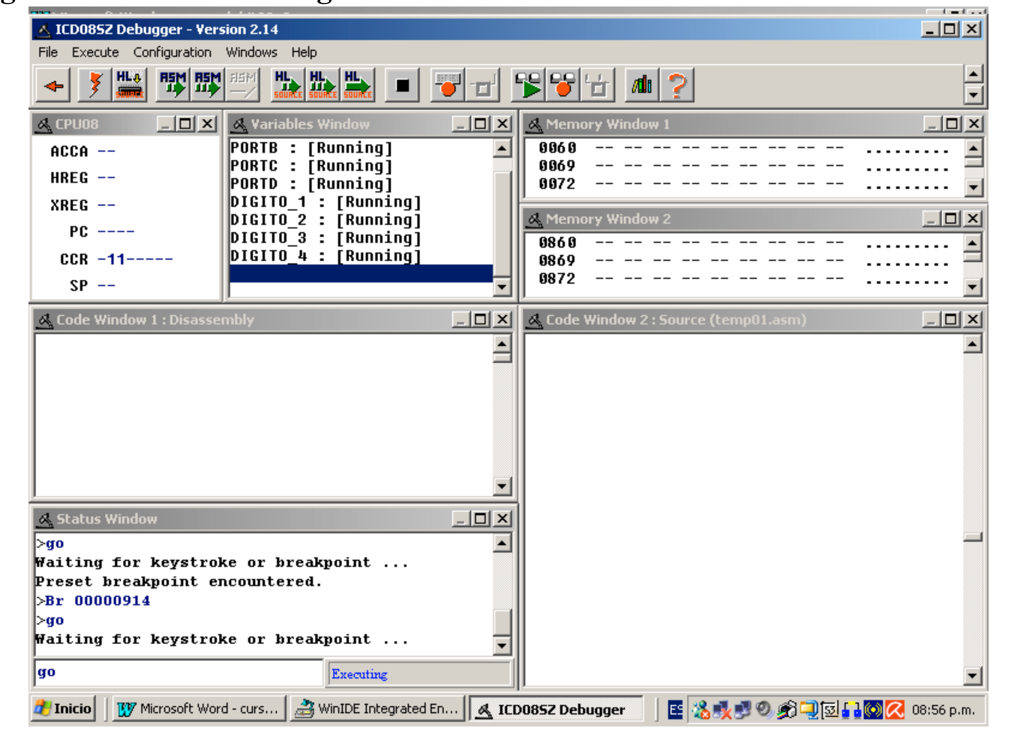


Luego corremos el programa "TEMP01" en tiempo real por medio del botón **"Go" (flecha verde entera)** disponible en la barra de herramientas del emulador.

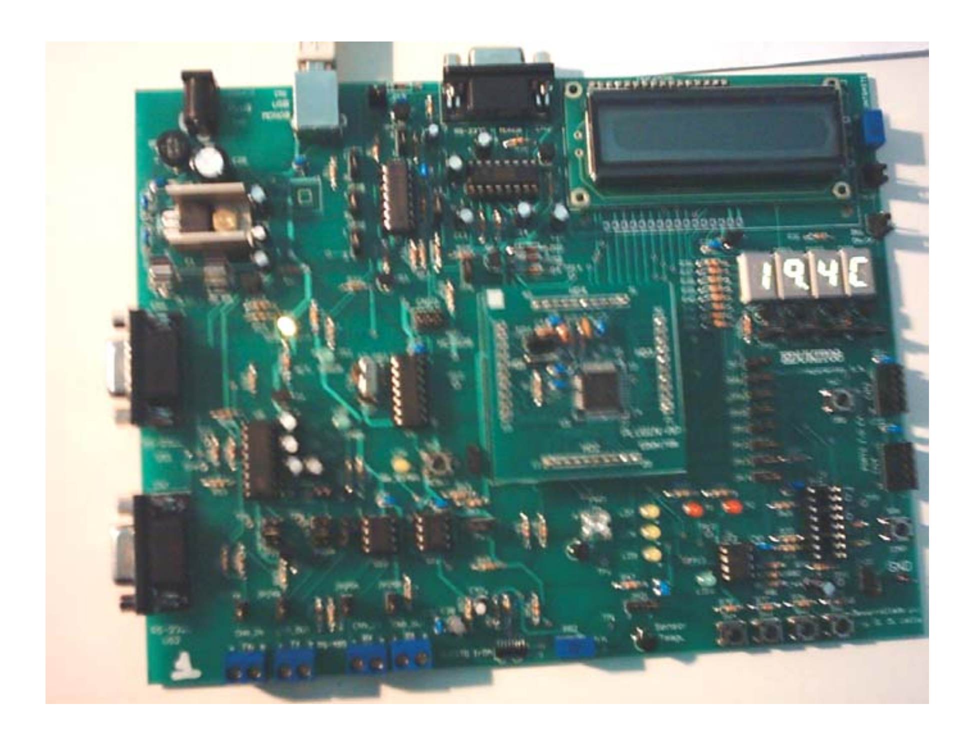
Luego de unos instantes el programa se detendrá en el punto de parada que previamente habíamos establecido devolviendo valores de variables y registros que fueron modificados por la "corrida" del programa en tiempo real **(ver figura).**



Pantalla de Emulación con el programa detenido en el Break Point y las variables y registros actualizados luego de la detención.



Pantalla del sistema "corriendo" (running) sin encontrar el Breakpoint.



Sistema "EDUKIT08" corriendo la aplicación "TEMP01" durante la Emulación en Tiempo Real con el ICD08SZ del WinIDE.

Hasta aquí hemos llegado con las demostraciones prácticas de los distintos aplicativos disponibles en el entorno **WinIDE** y como utilizarlos, sugerimos al usuario leer detenidamente el documento ***"WinIDE User Manual.pdf "*** (contenido en el CD ROM de instalación del sistema) que es manual del usuario del entorno WinIDE para obtener las máximas prestaciones de esta herramienta de software. Muchas son las posibilidades de este entorno, solo la práctica y la atenta lectura del manual del usuario dará al lector el control total de dicho entorno.

Sistema *"CodeWarrior"* for HC(S)08, de Freescale Semiconductor.

El sistema ***"CodeWarrior"*** es un entorno integrado de trabajo (IDE) de uso profesional que reúne en un mismo entorno un compilador de código Assembler y código "C", un linkeador y un debugger de código assembler fuente / código "C", y que puede ser ampliado según las necesidades del usuario.

El entorno de trabajo ***CodeWarrior*** es básicamente el mismo desde microcontroladores muy pequeños de 8 Bits hasta microcontroladores complejos de 32 Bits. Este aspecto es muy interesante ya que "acostumbra" al diseñador con microcontroladores a moverse dentro de un entorno único y a no encontrarse con "un mundo nuevo" cada vez que efectúa la migración de un proyecto con un microcontrolador de 8 Bits a otro de 16 o 32 Bits.

Existen varias versiones del sistema ***CodeWarrior***, cada una de ellas soporta una serie de familias de microcontroladores de distinta complejidad y aplicaciones.

Por ejemplo, para movernos dentro del sistema didáctico ***"EDUKIT08"***, las versiones que se pueden utilizar son las siguientes:

- **CodeWarrior 5.0:** apto para MCUs de la familia HC908 y HC9S08, puede instalarse en PCs o Notebooks con sistemas operativos Windows 98SE / Me / XP y no requiere de grandes exigencias del hardware de la PC, con procesadores Pentium III o similar y 128 Mbytes de RAM es suficiente. La versión gratuita permite compilar sin límites código assembler y hasta 16 Kbytes en código "C".

- **CodeWarrior 5.1:** apto para MCUs de la familia HC908, HC9S08 y RS08, puede instalarse en PCs o Notebooks con sistemas operativos Windows XP o superior y requiere de procesadores Pentium IV o equivalentes y más de 256 Mbytes de RAM. Posee un muy buen nivel de actualizaciones y soporta muchas herramientas distintas. La versión gratuita permite compilar sin límites código assembler y hasta 32Kbytes en código "C".

- **CodeWarrior 6.x:** Son las versiones disponibles desde el 2007, aptos para MCUs de la familia HC908, HC9S08, RS08 y la nueva familia "Flexis" de 8 / 32 Bits. Pueden instalarse en PCs o Notebooks con sistemas operativas Windows XP o superior y requieren de procesadores Pentium IV o superiores y más de 512 Mbytes de RAM.
- Poseen un muy buen nivel de actualizaciones y soporta muchas herramientas distintas. La versión gratuita permite compilar sin límites código assembler y hasta 32Kbytes en código "C" para la familia HC9S08 y 64Kbytes para la familia Flexis HC9S08 / V1 ColdFire.

En esta sección trabajaremos con la versión 5.0 del CodeWarrior, ya que su forma de manejo es similar en todas las versiones .

La versión "5.0" ofrece ensamblado de código fuente en forma ilimitada (assembler) y provee capacidades de Debugging muy interesantes aún para programadores adelantados.

Esta herramienta poderosa, combina un Ambiente de Desarrollo Integrado de Alta perfomance (I.D.E) con:

- Simulación Completa del Chip y programación de la memoria FLASH desde el sistema ***EDUKIT08.***

- Un Compilador ANSI C (16K Bytes de código) altamente optimizado y un Debugger en nivel fuente C, Generación automática de código C con "Processor Expert" desde unidades.

Ejecutar una sección de programación o debugging con proyectos basados en entornos ***CodeWarrior*** **IDE** es tan simple como efectuar un doble "click" en el nombre del proyecto (el formato es ***"nombredelproyecto.mcp"***) desde el archivo almacenado. Comenzar un nuevo proyecto es un poco más complejo, pero los tutoriales, FAQs y guías rápidas de comienzo son fáciles de seguir y ayudan a construir un nuevo proyecto, usando "templates" pre-construidos, en muy poco tiempo.
(Ver www.freescale.com y seleccionar.....

"CodeWarrior For HC(S)08 Microcontrollers".)

El siguiente ejemplo ilustrará como programar y depurar código en un MCU HC908 desde un entorno ***CodeWarrior*** IDE.

Aquí se darán los pasos principales en la programación de la memoria FLASH con ***CodeWarrior*** en **modo monitor** y como comenzar una sección de **debug** (depuración).

Nuestro ejemplo consistirá en implementar un pequeño programa en lenguaje ensamblador que efectúe una interrupción en forma periódica cada “n” milisegundos basado en el uso del Timer en modo TOV (Timer Overflow). Este sencillo programa nos servirá como base para ejecutar un número de tareas más complejas en forma periódica de modo similar a como lo haría un sistema operativo más complejo.

Primeramente configuraremos al sistema ***EDUKIT08*** para trabajar con el MCU **MC908AP32CFBE** que es el microcontrolador disponible en la placa “PLUGIN_AP” que viene con el kit del sistema.

Utilizaremos la conexión **USB** entre el sistema de desarrollo y nuestra PC, alimentando a todo el sistema didáctico con **+5Vdc** provistos por la propia PC.

De esta forma la configuración final será:

JP1 ---- Placa "PLUGIN_AP" ---- **Posición 2-3** (oscilador externo 20 Mhz).

JP2A / JP2B / JP2C --- Placa Principal

Posición 2-3 ---- Uso del Puerto Serial USB "CN1" al puerto USB 2.0 de la PC.
(No usar fuente de alimentación Externa!!).

JP3 --- Placa Principal ---- **Posición 1-2** (Control de alimentación por DTR).

JP4 --- Placa Principal ---- **Posición 2-3** (+VHIGH en pin RESET)

JP5 --- Placa Principal ---- **Posición Cerrado** (Manejo del pin de Reset)

JP6 --- Placa Principal ---- **Posición Cerrado** (Manejo del pin de IRQ)

JP15 / JP16 --- Placa Principal ---- **Posición Abierto** (Display LCD OFF).

JP17/JP18/JP19/JP20 --- Placa Principal ---- **Posición Abierto**
(Display 7 Segmentos LEDs OFF).

Cabe recordar que como utilizaremos el puerto USB de nuestro sistema para comunicarnos, se deberá comprobar que se hayan instalado los drivers necesarios, provistos con la herramienta, para que la PC o Notebook utilizada interprete como un puerto **"COM Virtual"** al puerto USB físico, ya que en los entornos de trabajo integrados como el ***CodeWarrior*** o ***P & E WinIDE*** utilizaremos la opción "COM" como medio válido para comunicarnos (ver sección **"Poniendo en Marcha el sistema"** del manual del usuario del sistema ***EDUKIT08***).

Configurado nuestro sistema, procederemos a iniciar nuestro programa en el sistema ***CodeWarrior 5.0*** efectuando los siguientes pasos:

Al ejecutar el ***CodeWarrior* IDE**, se nos abrirá una ventana de opciones como se ve en la **figura 1.**

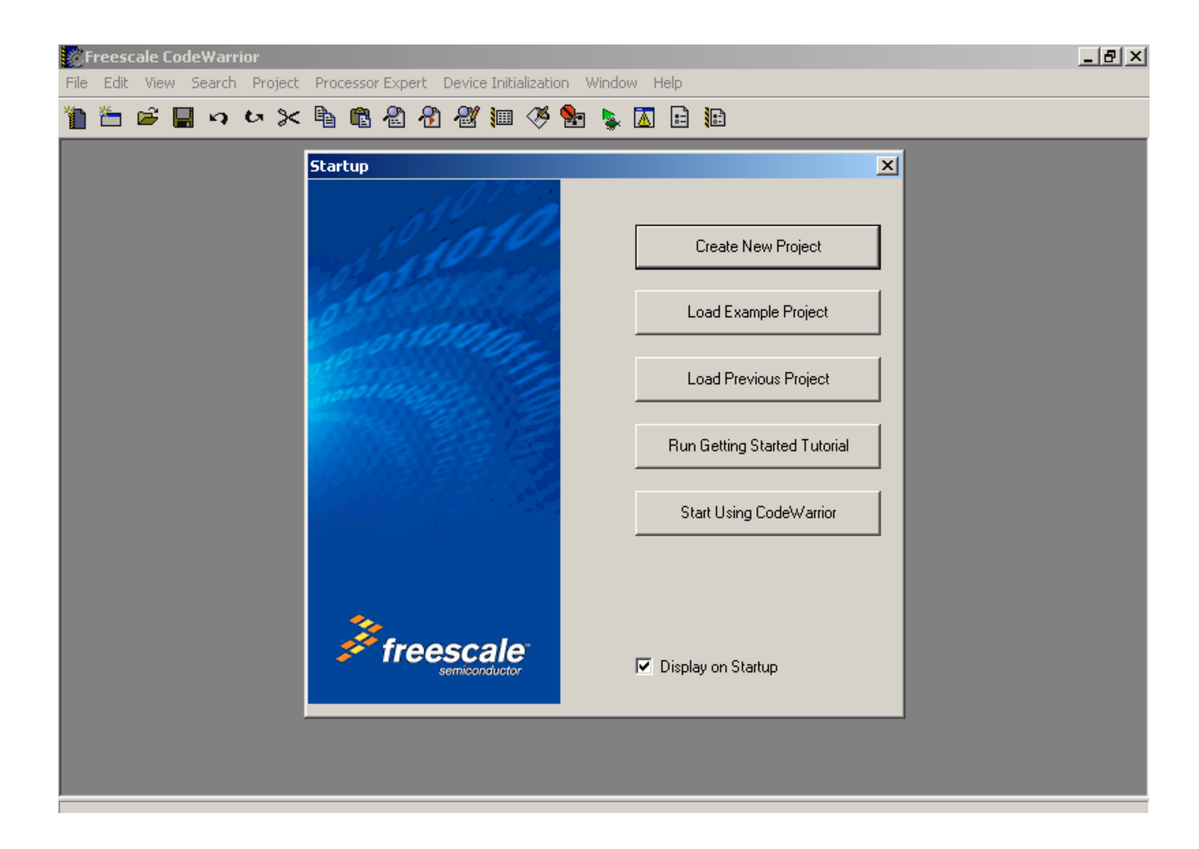
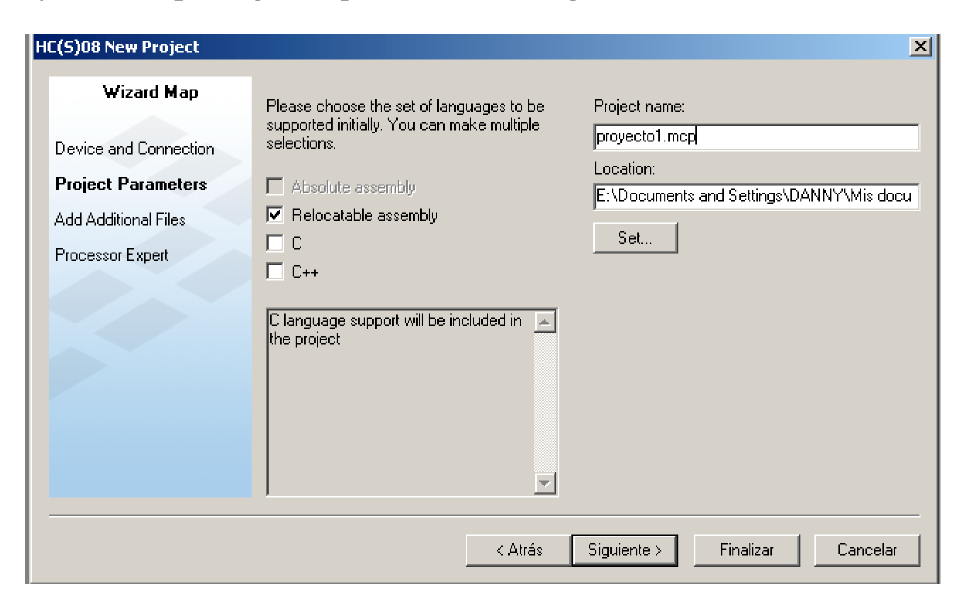


Figura 1.- Pantalla "Startup" con opciones de ayuda.

Elegiremos la opción **"Create New Project"** para armar nuestro nuevo proyecto .....

1) Se ingresará en la pantalla de configuración del proyecto donde elegiremos generación de lenguaje **ASSEMBLY**, y nombraremos a nuestro proyecto con el nombre **"proyecto1.mcp",** según se puede ver en **la figura 2.**



En la siguiente pantalla configuraremos la familia y dispositivo en particular a utilizar para nuestro proyecto (**MC908AP32CFBE**) según se puede ver en la **figura 3.**

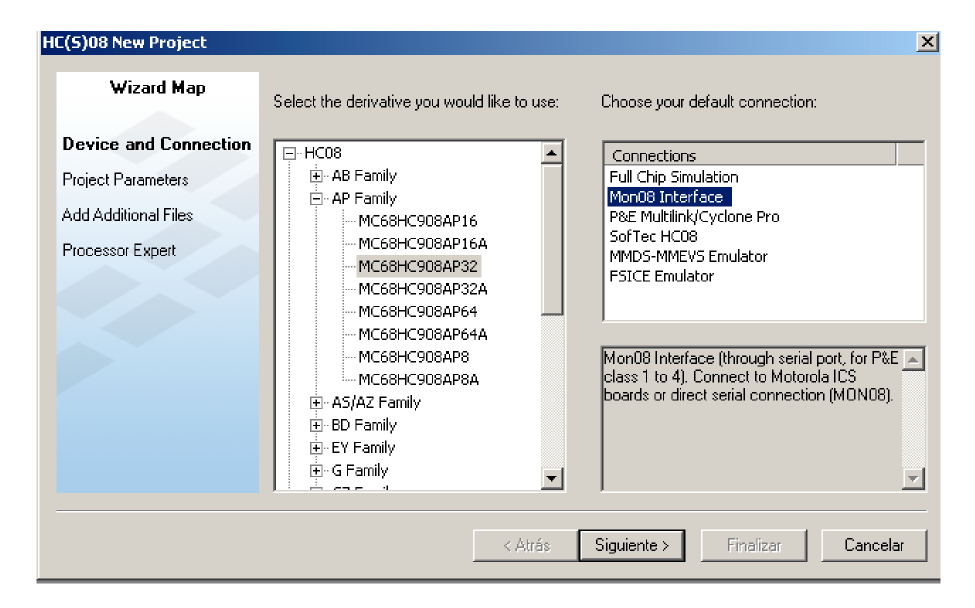


Figura 3.- Pantalla de configuración de Familia, dispositivo y tipo de conexión.

Como se observa en la **figura 3**, se ha elegido la familia HC908, dispositivo **MCHC908AP32CFBE** y en cuanto a la conexión con la herramienta debe elegirse la opción **"Mon08 Interface"** ya que es la opción universal de conexión para los sistemas de desarrollo como los ***EVAL08QTY, FLASH_POD, EDUKIT08*** y toda otra herramienta que no figure explícitamente en el listado de conexiones.

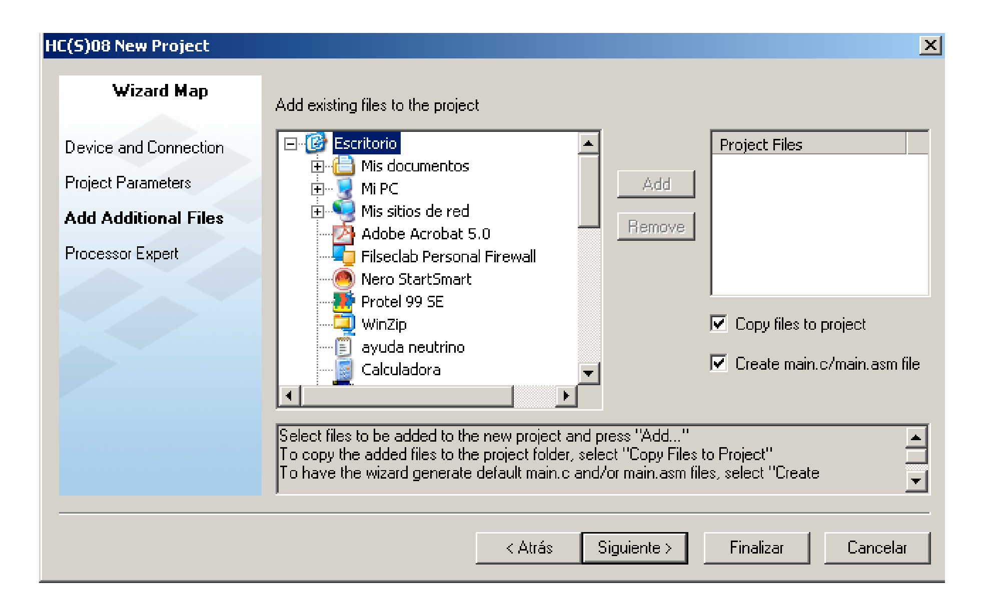


Figura 4.- Pantalla de adición de archivos al proyecto.

Al hacer click en el botón **"siguiente"** pasamos a una pantalla (**Figura 4.-)** que nos permite adicionar cualquier archivo al proyecto, para incluirlo en nuestro trabajo. En nuestro caso, saltaremos esta opción e iremos a la próxima pantalla.

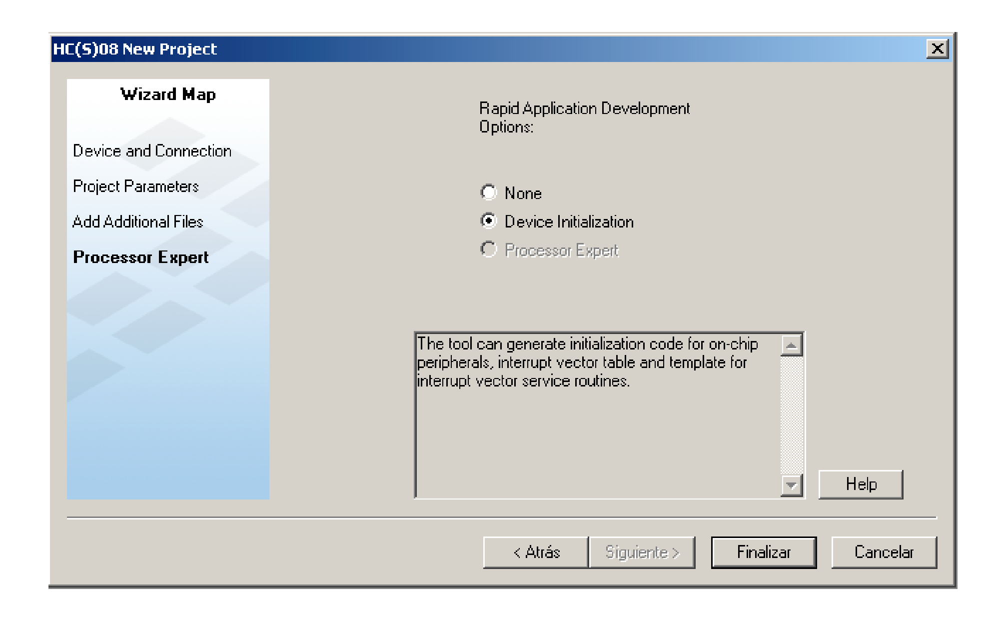


Figura 5.- Pantalla de elección o no de generación de código asistido (Processor Expert).

En la pantalla que se observa en la **figura 5.-**, se puede elegir la generación de código de inicialización de los distintos periféricos asistida o no. Nosotros elegiremos utilizar la generación de código asistida, por medio del aplicativo ***"Processor Expert",*** que nos irá guiando paso a paso en la inicialización de los distintos periféricos del MCU elegido.

Al hacer click en el botón **"Finalizar",** se generarán todos los archivos del proyecto, se lanzará la pantalla principal de trabajo del mismo y se podrá ver una interface gráfica con los pines y los distintos módulos que constituyen el MCU **(Figura 6.-).**

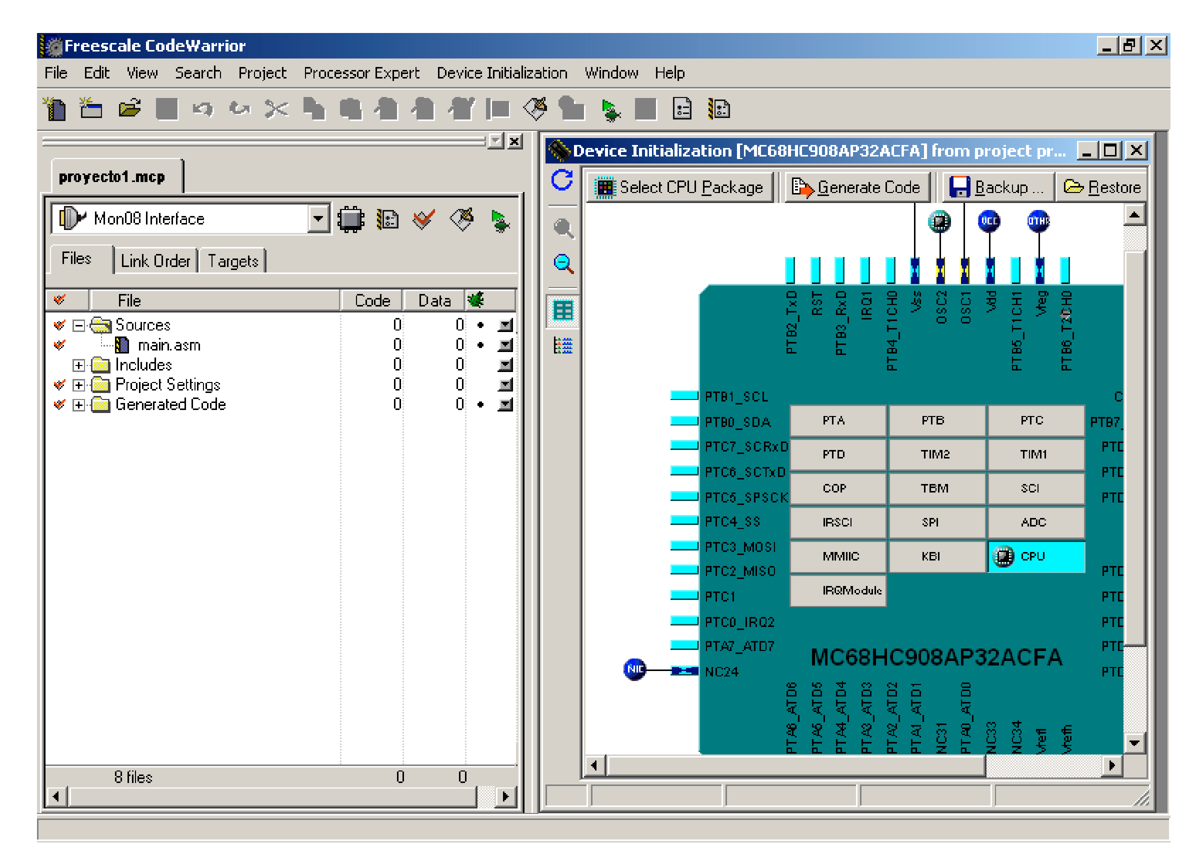


Figura 6.- Pantalla principal del proyecto e interface gráfica de generación de código (Processor Expert).

Hacer "click" en el icono ***"Select CPU Package"*** y elegir el encapsulado ***"MC68HC908AP32CFB"*** que es el correcto utilizado en nuestro kit.

Ahora nos queda generar el código de inicialización del Timer para producir una interrupción periódica que será la base de nuestro sistema de disparo de tareas, inicializar los puertos I/O, los registros de configuración, etc., etc.
Para hacer esto, usaremos el generador de código asistido "Processor Expert" haciendo click primeramente en el modulo CPU para configurar el Clock del sistema y otros aspectos como se observa en la **figura 7.-**

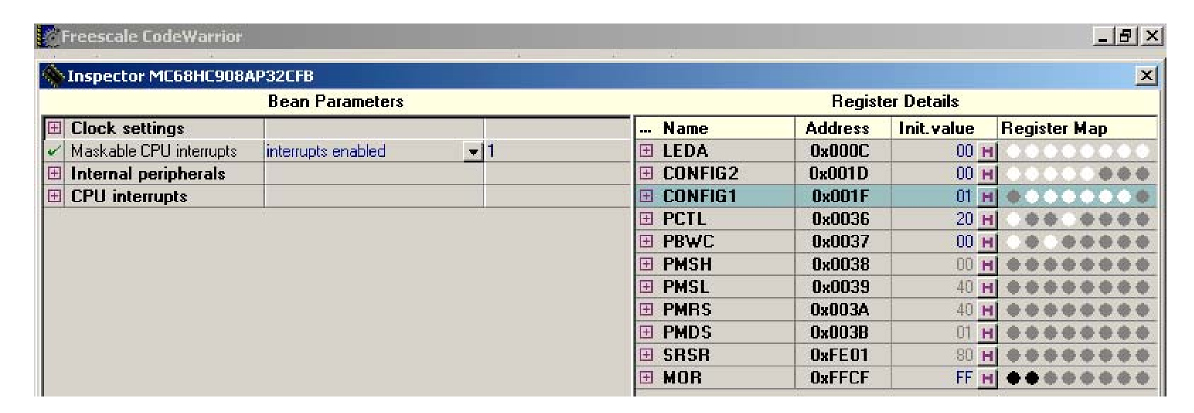


Figura 7.- Pantalla del módulo de CPU.

Se configura el **módulo de CPU** para:

- Clock ---- Externo ---- **20,000Mhz** (lo inyectará ***EDUKIT08*** por pin OSC1).
- LVI ----- LVI en +VDD habilitado / LVI en Reset habilitado / LVI en +Vreg habilitado / LVI en modo STOP deshabilitado.
- PLL Clock ---- **Deshabilitado.**
- CPU mode Selection ----- **"Monitor Mode".**
- PTB0 pin level ------ **"Logical 1".**
- Interrupciones Habilitadas.
- Vector de Reset apuntando a la etiqueta **"_Startup"**
- SWI **deshabilitada.**

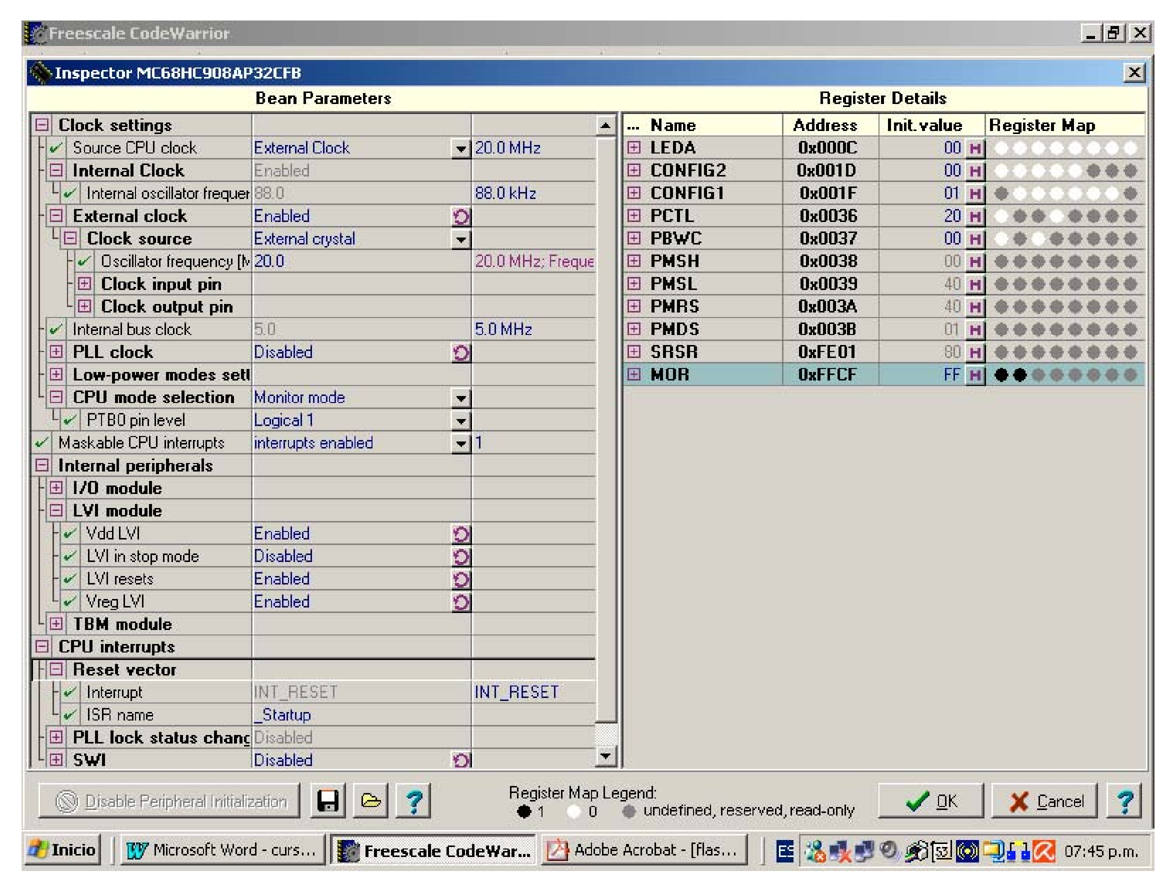


Figura 8.- Pantalla con los detalles de configuración del CPU.

A continuación procederemos a configurar el **módulo de Timer (TIM)** ingresando al mismo como muestra en la **figura 9.-**

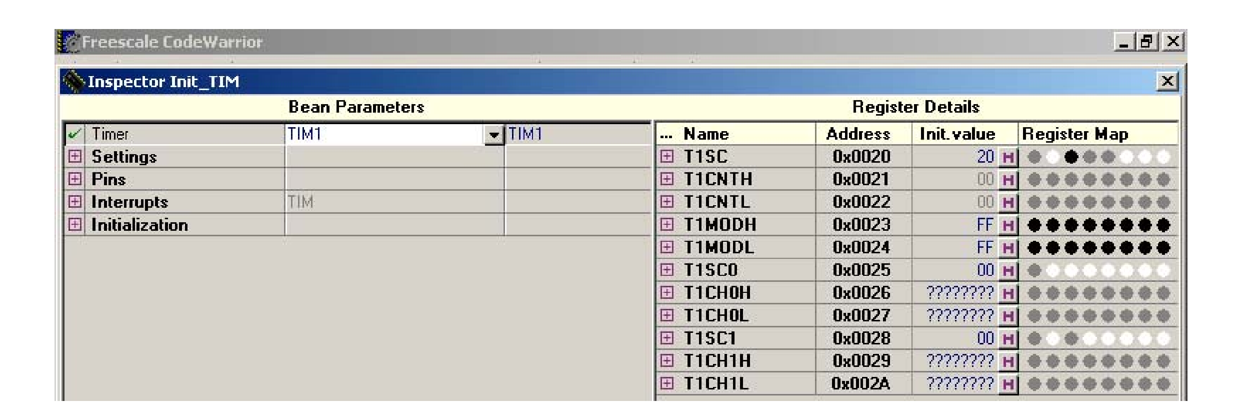


Figura 9.- Pantalla del Módulo de Timer.

Ahora es el turno de configurar el módulo del TIMER según lo siguiente:

- Prescaler = 32 ----- FBUS = 5,0000 MHz.
- Período del timer = 100 ms
- Modo de funcionamiento ----- Timer Overflow Interrupt (INT_TIMOvr).
- Overflow Interrupt = habilitado.
- Nombre de la interrupción = **isrINT_TIMOvr**
- Inicialización = Comienzo de la cuenta (arranque del timer).

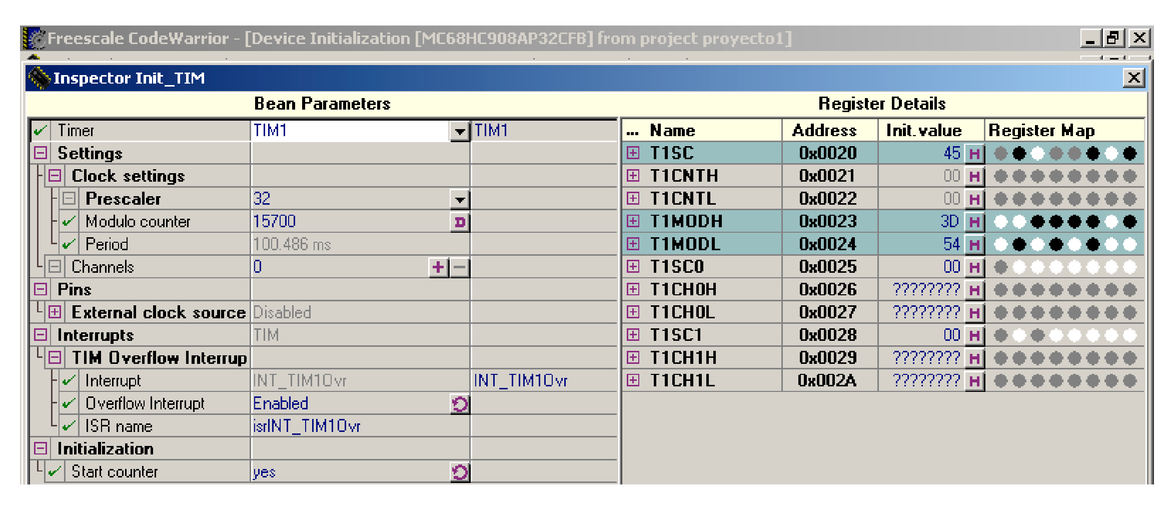


Figura 10.- Pantalla configuración del TIMER

Una vez que se ha configurado el módulo de **TIMER**, procederemos a configurar los puertos I/O según lo siguiente:

PORTA ---- DDRA = % 11110110 ---- PTA = % 00000010.

PORTB ----- DDRB = % 11100110 ---- PTB = % 00000110.

PORTC ----- DDRC = % 01110111 ---- PTC = % 01010000.

PORTD ----- DDRD = % 00001111 ---- PTD = % 00000000.

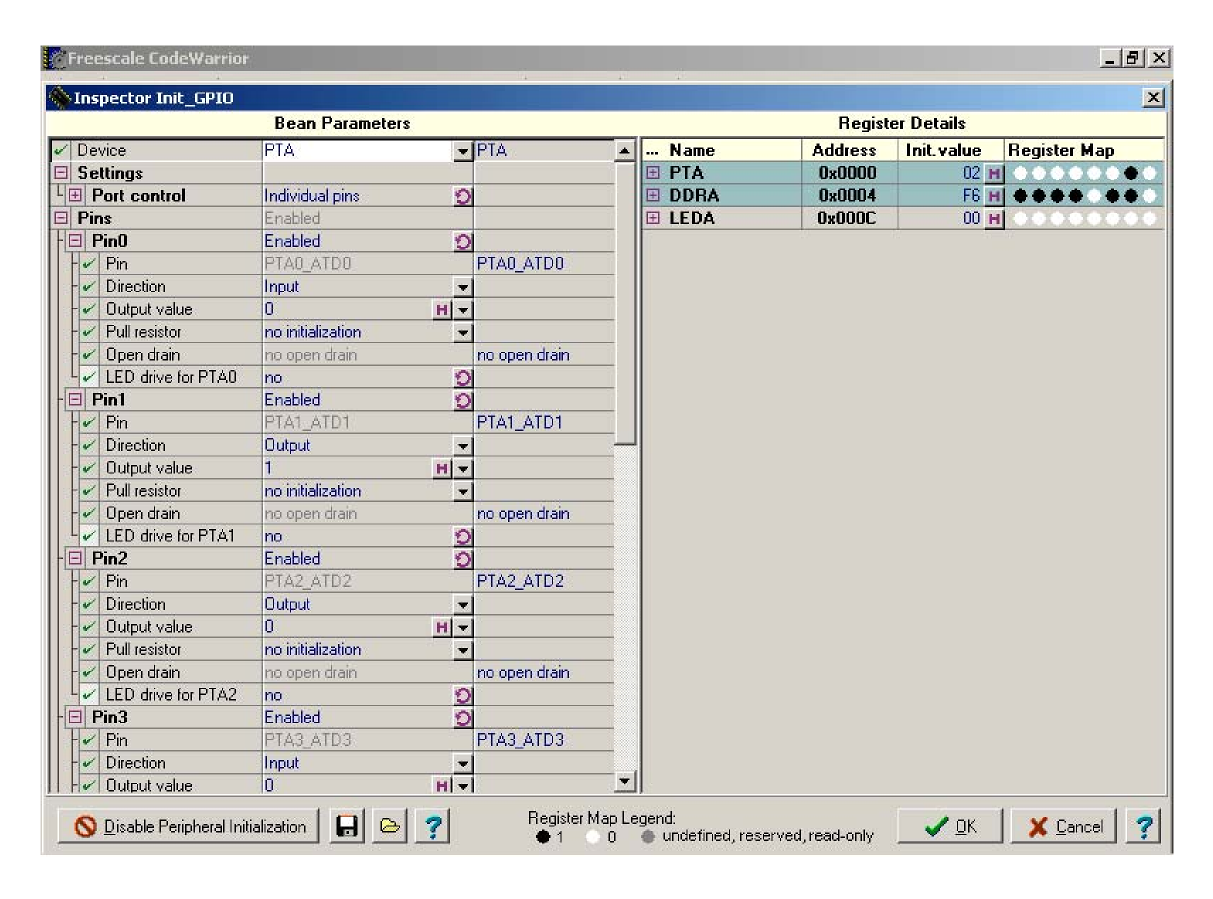


Figura 11.- Pantallas de configuración de puertos (PORTA / PORTB / PORTC / PORTD).

Si luego se presiona el botón **"Generation Code"** , el generador de código del ***Processor Expert*** generará código y nos mostrará una ventana explicando los pasos a seguir para incorporarlo efectivamente al resto del programa.

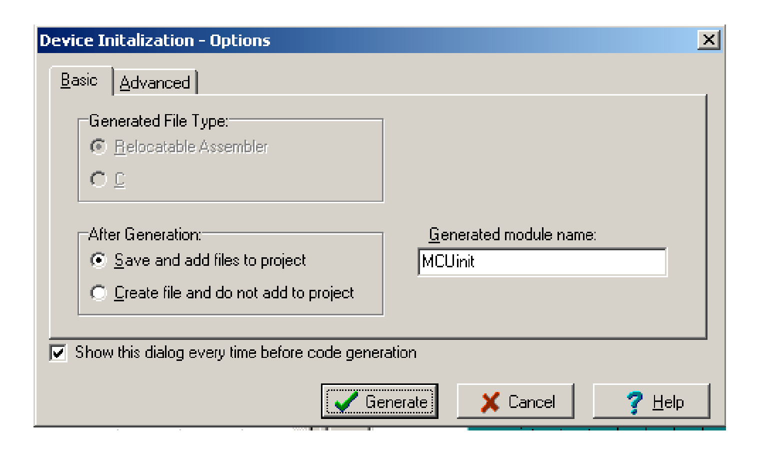


Figura 12.- Pantalla de generación de Código que producirá archivos bajo el Nombre "MCUinit" para inicializar el MCU.

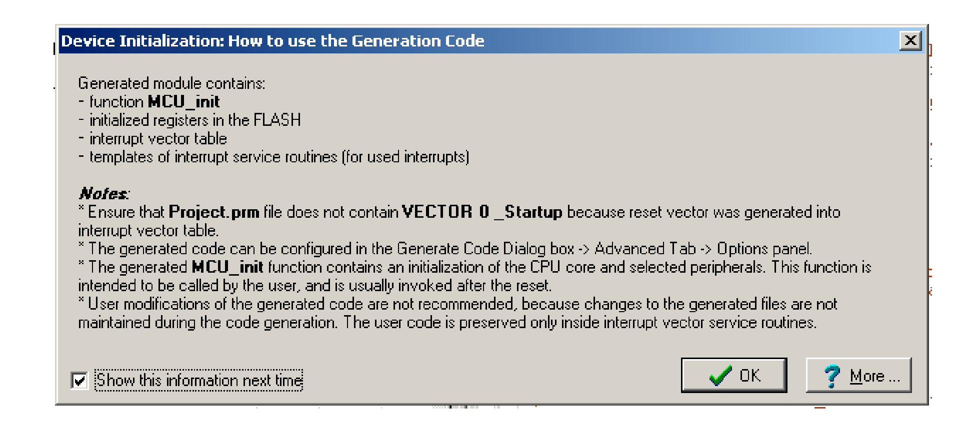


Figura 13.- Pantalla de ayuda para integrar el código generado al proyecto.

Según lo sugerido por la ventana de ayuda una vez generado el código, procedemos a comentar y descomentar lo siguiente (solo para el CodeWarrior 5.0, en otras versiones solo verificar que así suceda):

- Comentar en el archivo **"Project.prm"** la línea --- **VECTOR 0 _Startup /* Reset vector ......**
  Con una barra cruzada y asterisco **(/*)** delante de la sentencia y luego salvarlo.

- Descomentar en el archivo **"main.asm"** la línea **"JSR MCU_int"** para que de esta forma en el programa principal se pueda invocar a la sub rutina **MCU_int** que inicializa al MCU.

Luego de realizar esas modificaciones sugeridas por el ***Processor Expert***, introduciremos nuestras líneas de código en la sub rutina de interrupción por Timer Overflow **(isrINT_TIMOvr)** para realizar, por ejemplo, un ***Toggle*** (inversión de estado) del puerto **PTA1 (LED 1 del EDUKIT08)** cada vez que atendamos la interrupción propiamente dicha. En este punto podemos poner todas nuestras tareas en forma de llamado a sub rutina que se irán ejecutando una a una cada **100 ms.**

```
     BCLR   T1SC_TOF,T1SC                    ;Limpio bit TOF del timer1

     BRCLR  PTA_PTA1,PTA,OFF_PUERTO          ;Invierto estado puerto PTA1
     BCLR   PTA_PTA1,PTA                     ;Enciendo LED 1 del EDUKIT08
     BRA    FIN_TIMOvr                       ;salto a fin rutina


OFF_PUERTO:

     BSET  PTA_PTA1,PTA                      ;Apago LED 1 del EDUKIT08


FIN_TIMOvr:

     RTI
```

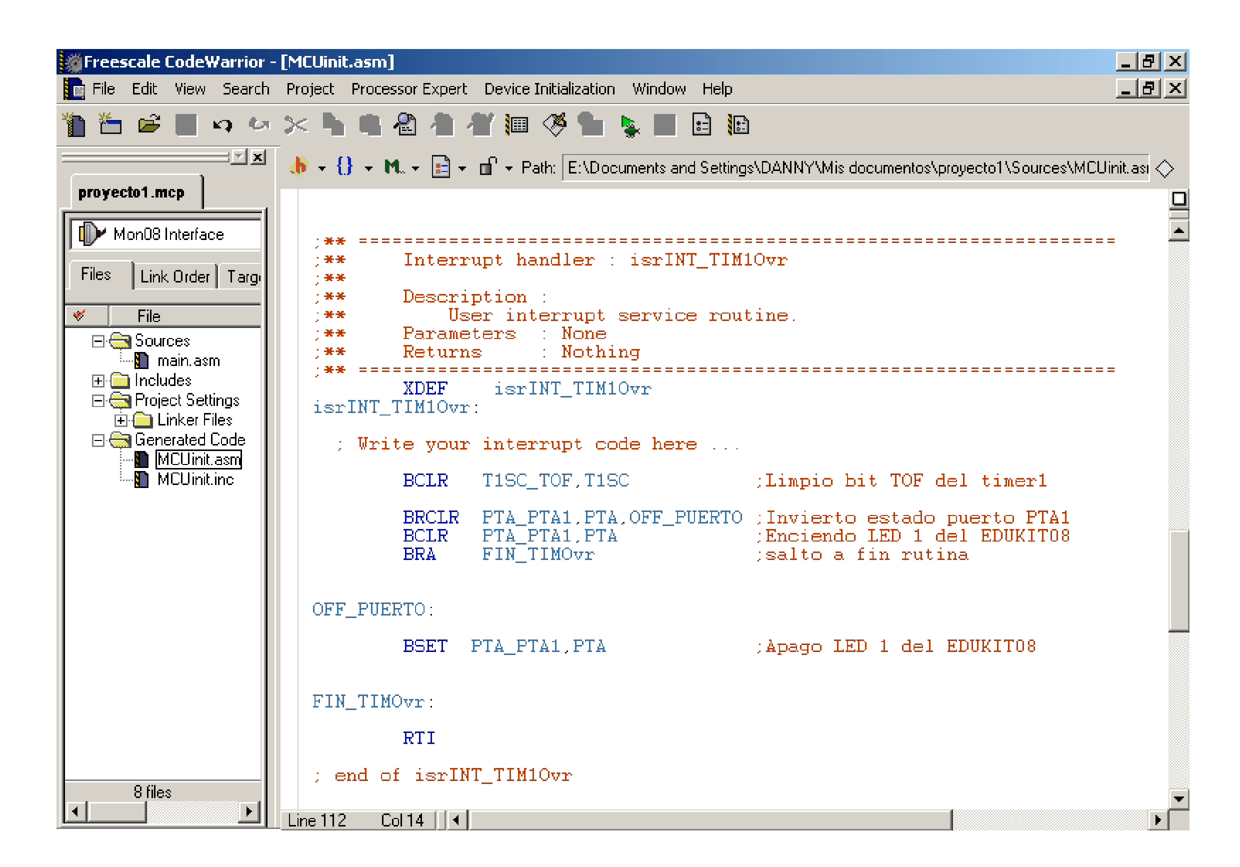


Figura 15.- Agregado de nuestras líneas de código en la rutina de manejo de la interrupción por Timer Overflow (isrINT_TIMOvr).

Una vez introducido nuestro código, debemos compilarlo haciendo click en el botón ***"Make"*** en la barra de proyecto o en la barra de herramientas general. Si no hemos tenido algún error de compilación estaremos ya en condiciones de pasar a la etapa de **EMULACION EN TIEMPO REAL** de nuestro programa.

Para realizar ello, primero deberemos establecer una conexión entre el ***CodeWarrior 5.0*** y nuestro sistema de desarrollo ***EDUKIT08*** que iremos configurando a lo largo de las siguientes pantallas luego de hacer click en el botón **"Debugger"** **(fecha verde en la barra de proyecto).....**

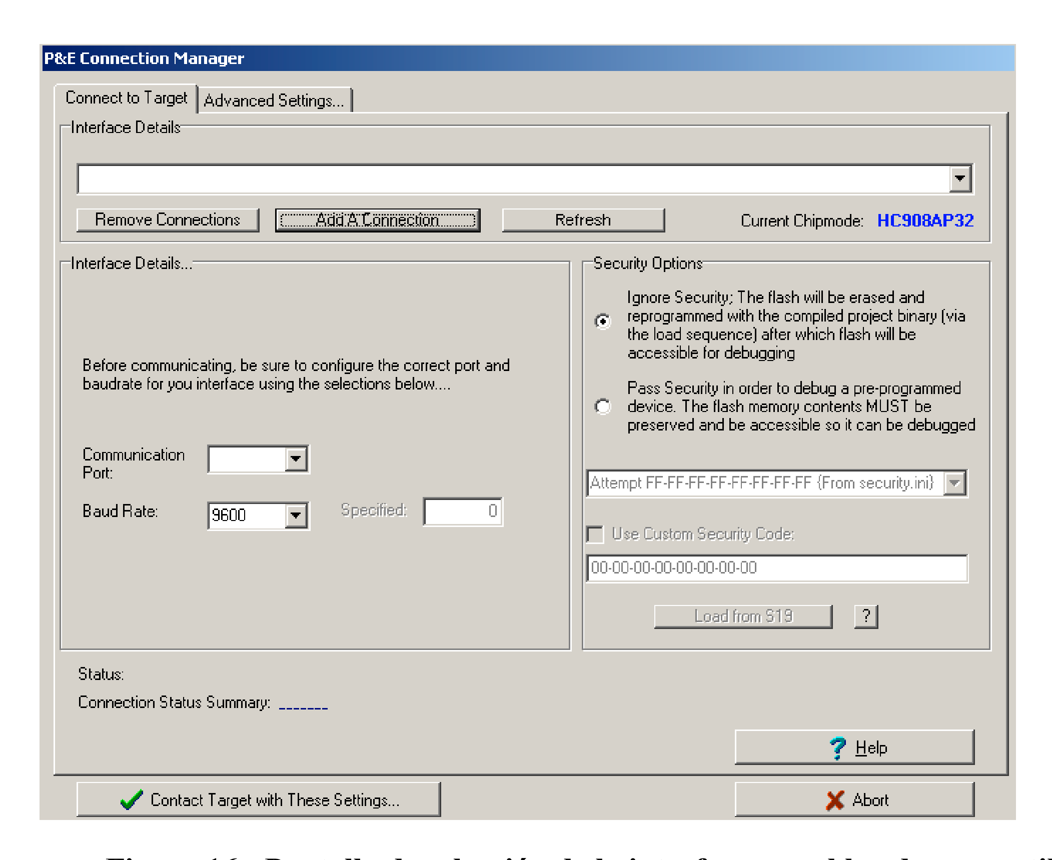


Figura 16.- Pantalla de selección de la interface con el hardware a utilizar.

En la ventana **"Interface Selection"** elegimos la opción **"Class 1 – ICS Board with processor installed"** y hacemos click en el botón **"ok".**

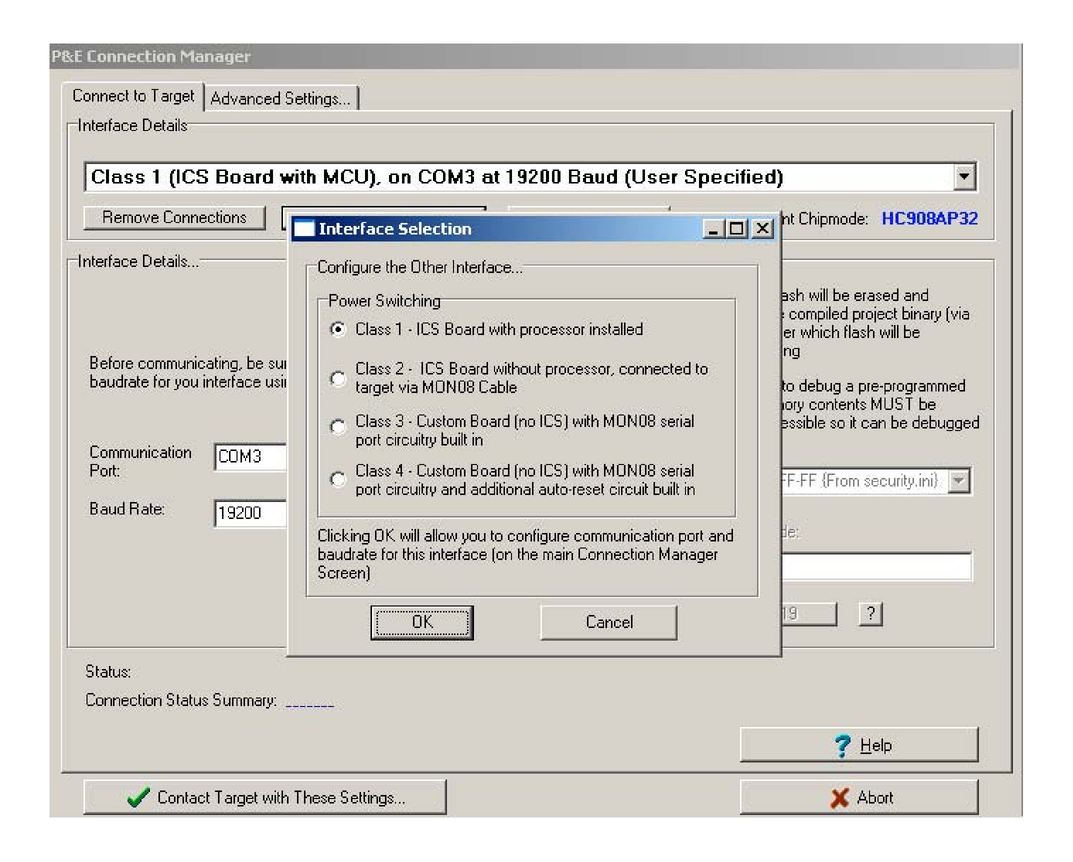
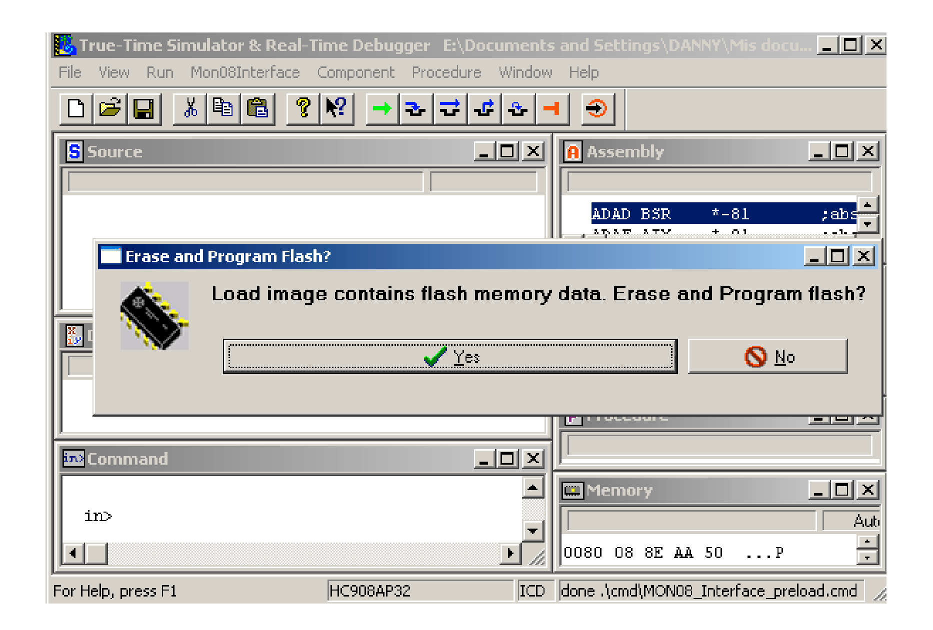


Figura 17.- Pantalla de manejo de la conexión con el hardware (FLASH_POD).

Luego configuramos la siguiente pantalla **eligiendo el número de puerto COM** en el que esté asignado el puerto **COM VIRTUAL** utilizado por ***EDUKIT08*** para la conexión USB SISTEMA – PC, en nuestro caso es el **COM3** y le asignaremos un Baud Rate de **19200 Bps** de acuerdo a lo configurado en nuestro sistema anteriormente.

Como se podrá observar en la figura, también se configurará la opción de **borrado** y **grabación** de la memoria FLASH del MCU en forma previa y automática cada vez que se quiera entrar en el modo de **Debugging** (Emulación en Tiempo Real) ya que es la condición necesaria para que cualquier HC908 pueda trabajar como una verdadera herramienta de desarrollo.

Hacemos click en el icono **"contact target with these sedttings...."** para establecer la comunicación con la placa ***EDUKIT08*** y entrar al entorno de Debugging propiamente dicho.



Una vez que nuestro sistema sortea las etapas de seguridad con éxito, nos aparecerá una ventana **(Erase and Program Flash?)** preguntando por el borrado y grabación de la FLASH antes de ingresar al modo de Debugging propiamente dicho. Haremos Click en el icono **"Yes"** para proceder a borrar y grabar nuestro programa en la memoria flash e ingresar al modo Debugging.

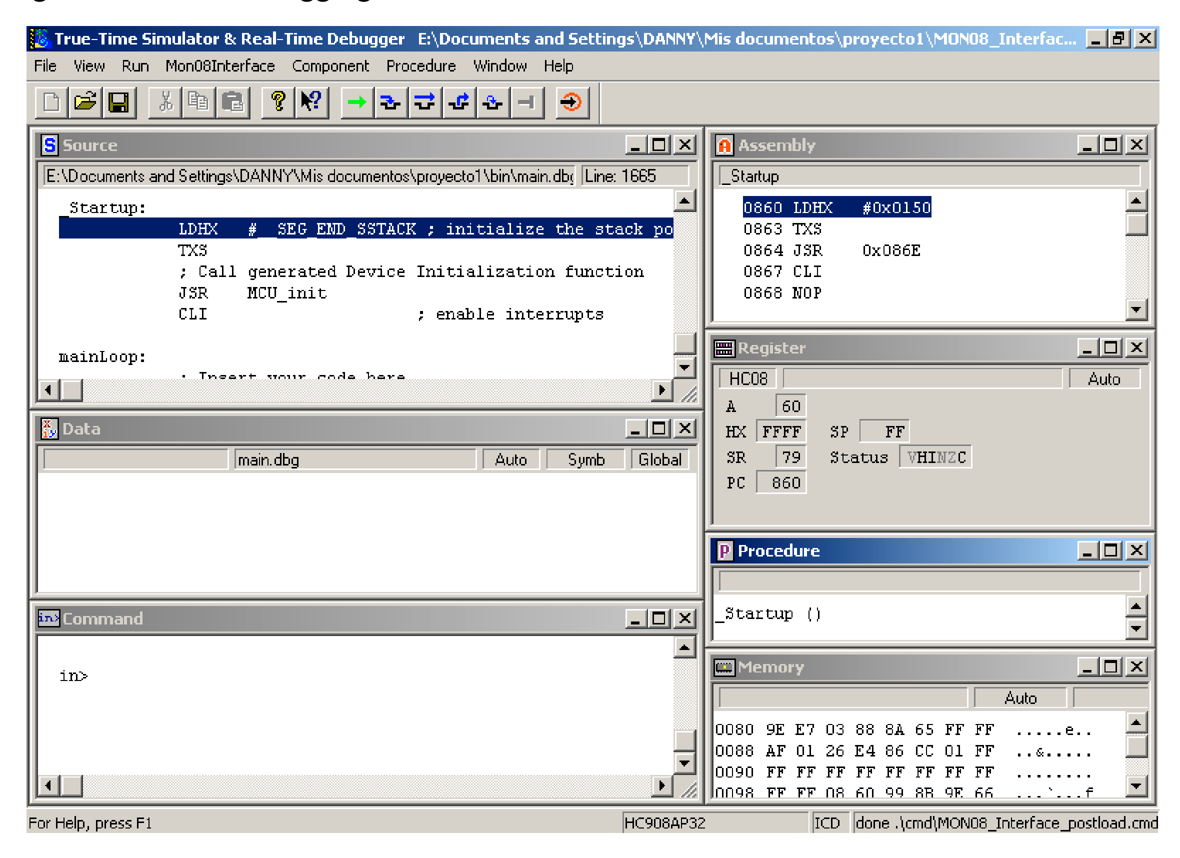


Figura 18.- Pantalla de Debugging (Emulación en Tiempo Real).

Pues bien, ya tenemos la pantalla principal de **Debugging (Emulación en Tiempo Real)** y solo nos resta correr nuestro programa haciendo Click en el icono con la **“flechita verde” (Run / Continue)** para poder ver **la señal cuadrada de 200 ms de período que obtendremos en el puerto PTA1** de nuestro sistema didáctico, según nos muestran las **figuras 19 y 20.**

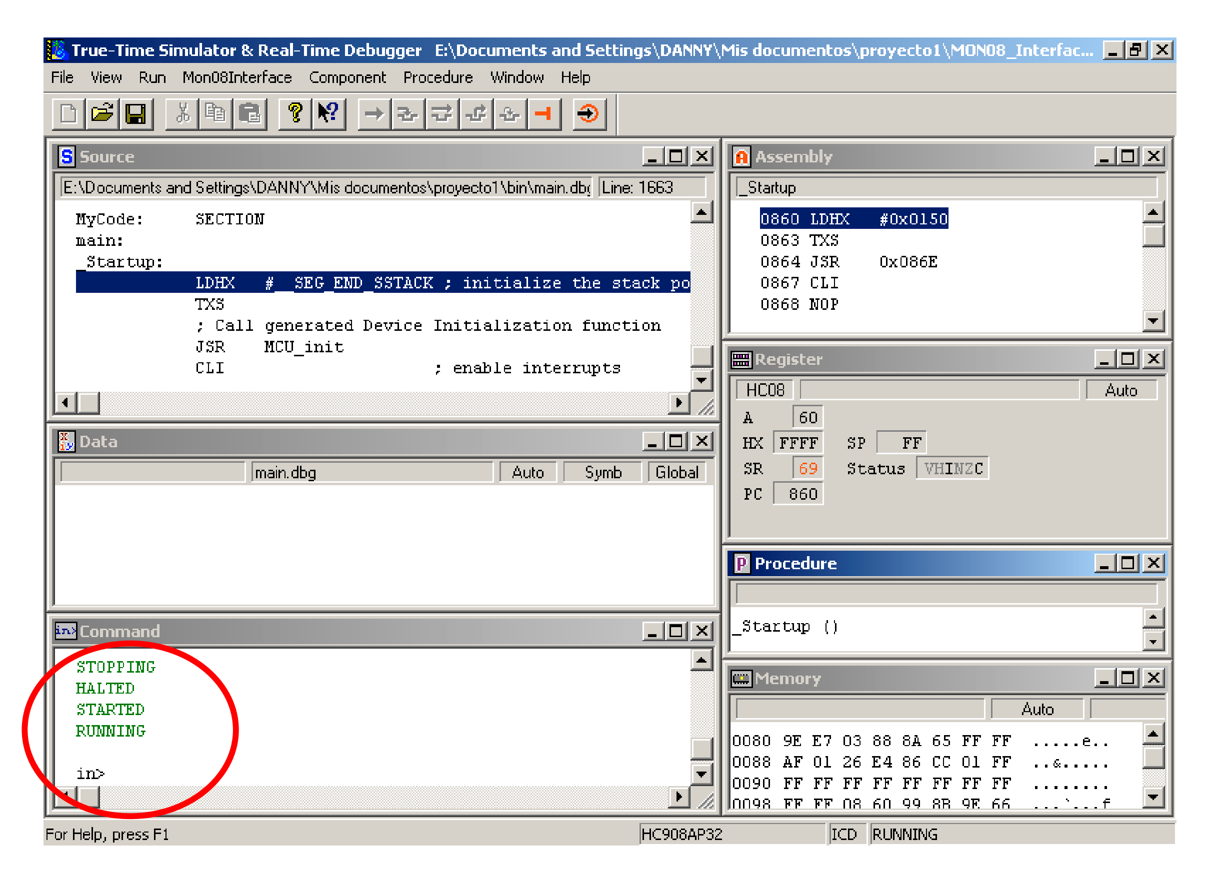


Figura 19.- Sistema “Corriendo” en Tiempo Real (Debugging).

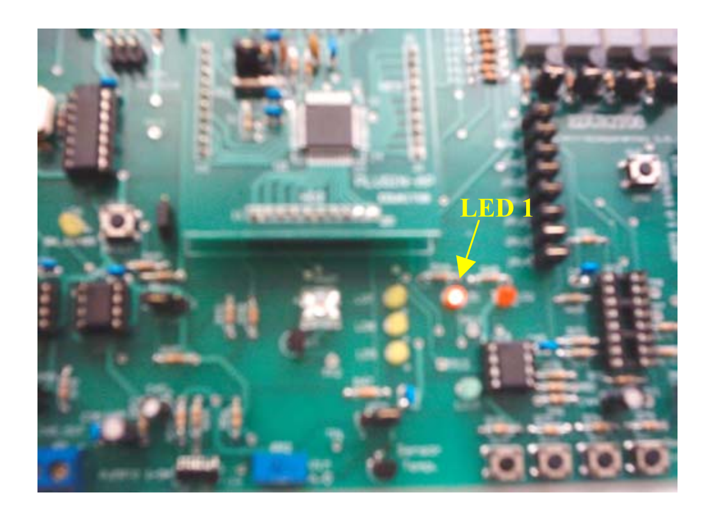


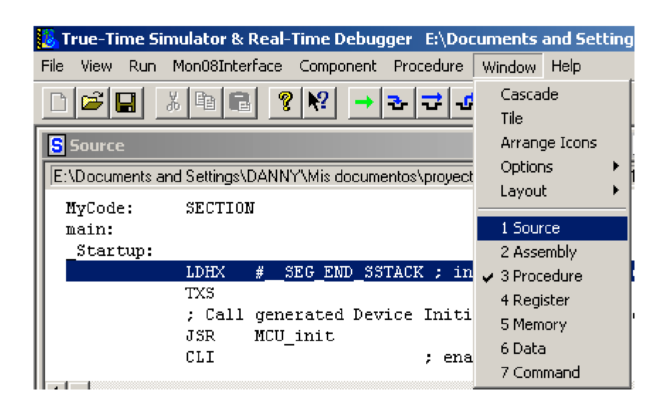
Figura 20.- Led “LD1” parpadeando con un período de 200 ms.

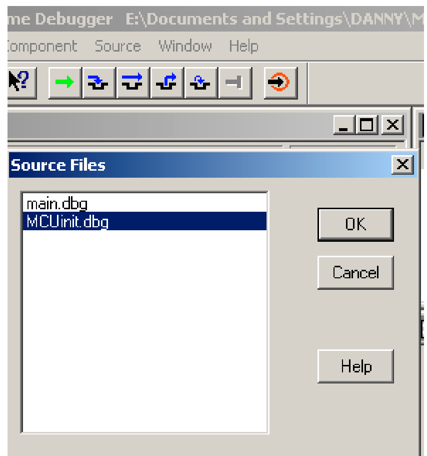
Para detenerlo solo tendremos que hacer "click" en el botón ***"Halt"*** de la barra de herramientas y se nos abrirá una pequeña pantalla que nos dirá que no tiene comunicación con el MCU bajo depuración, esto es lógico debido a que, como se vio en capítulos anteriores, la familia HC908 tiene un modo "monitor" que solo devuelve el control a la PC si el Program Counter (PC) pasa por la dirección donde se había colocado un "Breakpoint", y hasta ahora en nuestro ejemplo, no se ha colocado ninguno. Esta pantalla nos da varias opciones, la aconsejable es la opción ***"RESET"*** que produce un reset del MCU y con ello "restablece" el control del mismo en el modo monitor (Debugger).

Para aprovechar realmente la posibilidad de **Emulación en Tiempo Real** (Debugger), es necesario incluir un "Breakpoint" en algún punto de interés de nuestro programa para "detener" la ejecución normal del mismo en ese punto y poder rescatar los valores de las variables a examinar justo en ese momento.

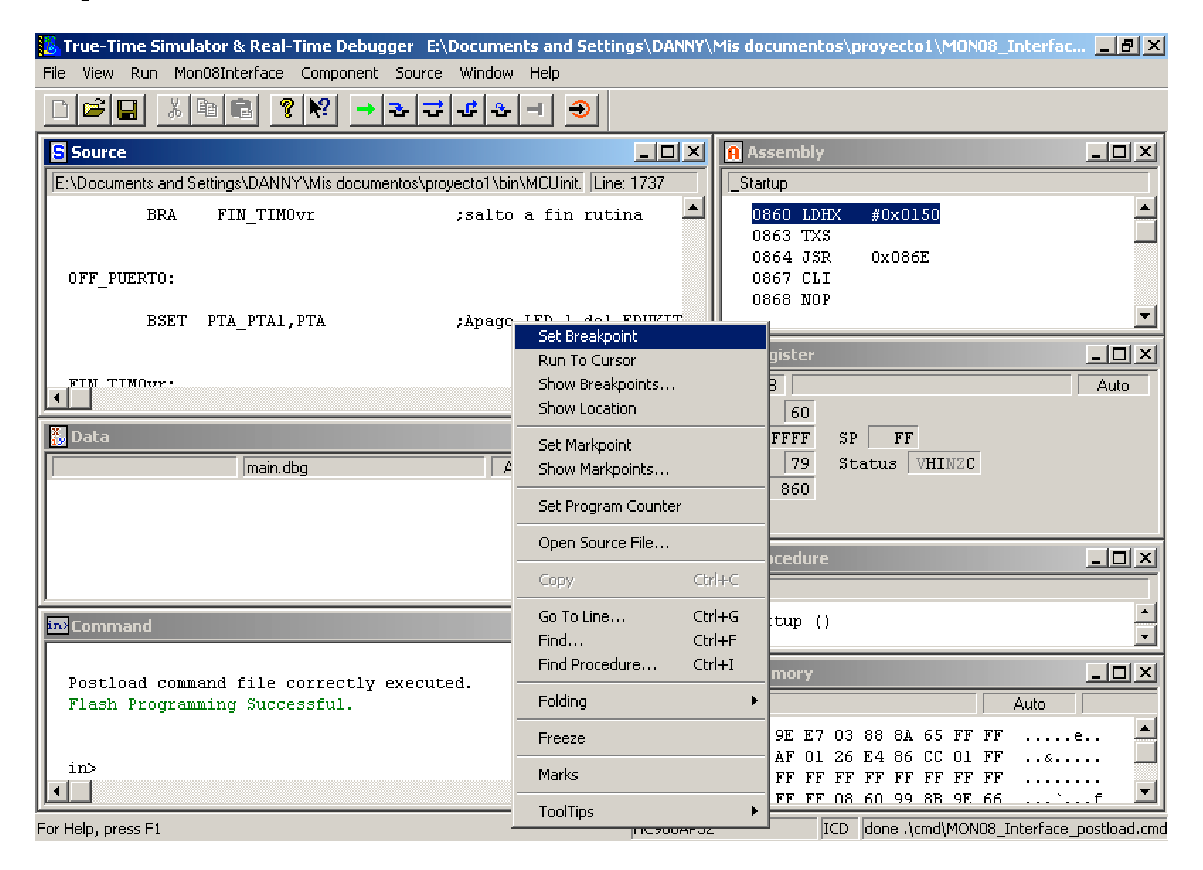
Por ejemplo, en nuestro ejemplo anterior, se introducirá un breakpoint en las líneas de código que se han agregado en la interrupción por "Timer Overflow", justo en la línea donde se apaga el LED "LD1".

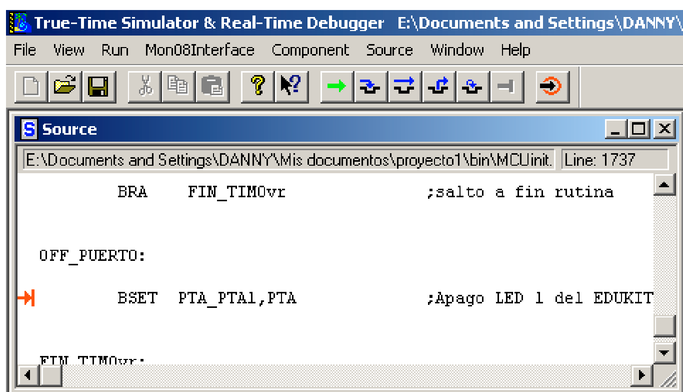
Para ello, se hará "click" en la opción ***"Source"*** de la solapa ***"Window"*** de la barra de herramientas del Debugger, para seleccionar y mostrar las líneas de código del archivo ***"MCUinit.dbg"*** que es el que contiene las líneas agregadas en la interrupción por Timer Overflow, según se muestra en las siguientes figuras.......



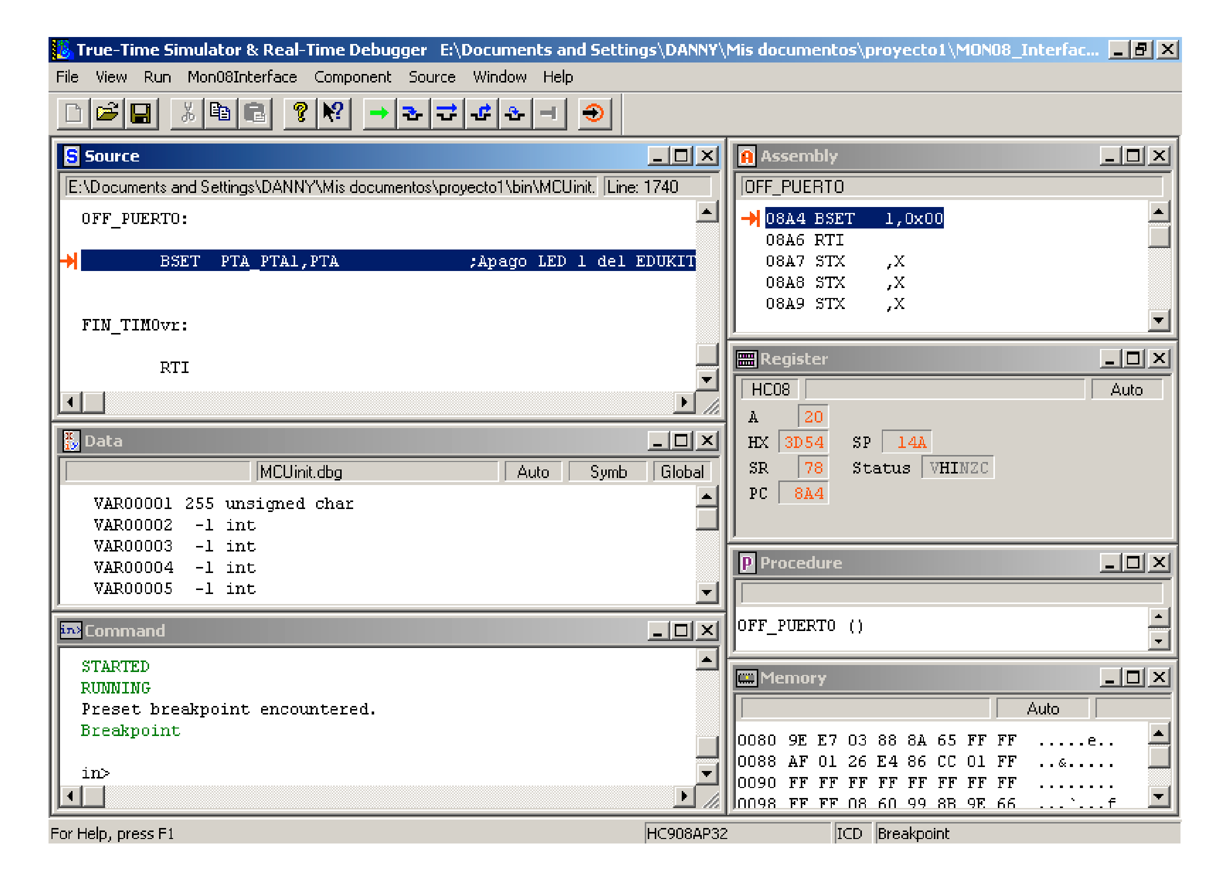


Una vez que en la ventana "Source" se pueden observar las líneas de código pertenecientes al archivo ***"MCUinit.dbg"***, se introducirá un "Breakpoint" en la línea elegida simplemente ubicando el puntero en dicha línea y haciendo "click" en el botón derecho del mouse se abrirá una ventana con distintas opciones, se elegirá ***"Set Breakpoint"*** haciendo "click" en la opción.......





A continuación, ya se está en condiciones de "correr" la aplicación con el Debugger haciendo "Click" en **la flechita Verde** de la barra de herramientas......



Pantalla del Debugger con el programa "detenido" en el breakpoint .....

Hasta aquí se han visto los aspectos principales de uso del entorno integrado ***CodeWarrior,*** sugerimos al lector ampliar sus conocimientos por medio del uso de la opción ***"Help"*** que dispone el entorno y además consultar los numerosos manuales y tutoriales disponibles en el sitio web de Freescale (www.freescale.com/codewarrior).

En el CD ROM de instalación del sistema didáctico **"EDUKIT08",** también se puede encontrar una carpeta bajo el nombre ***"Bibliografía CodeWarrior"*** que contiene una gran cantidad de información útil para aprender con dicho entorno.

Fin ¡!